VOLUME 05 | ISSUE 07

digit FastTrack

YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

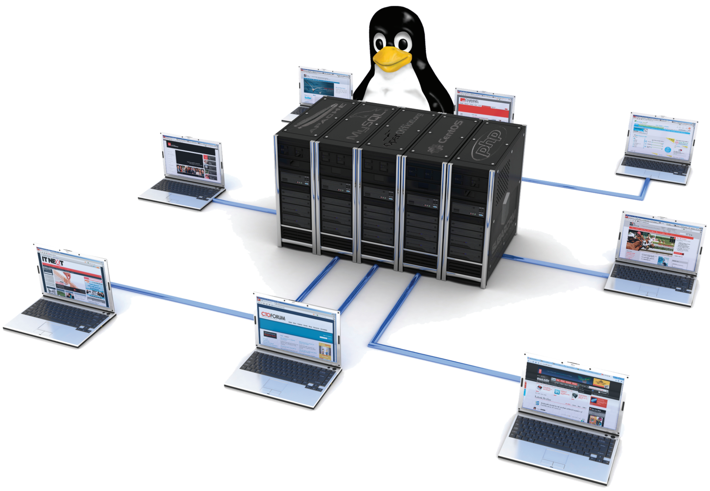
to Linux ADMINISTRATION

Fast Track to Linux Administration

A 9.9 Media Publication

digit
YOUR TECHNOLOGY NAVIGATOR
thinkdigit.com

Fast Track to Linux Administration



thinkdigit.com

CREDITS

The People Behind This Book

EDITORIAL

Editor	Robert Sovereign-Smith
Head-Copy Desk	Nash David
Writers	Nishith Rastogi and Rahil Banthia

DESIGN AND LAYOUT

Lead Designer	Vijay Padaya
Senior Designer	Baiju NV
Cover Design	Binesh Sreedharan

Contents

Introduction

Without going into a vi vs Emacs style debate at the very beginning, we'd like to state that it is an accepted opinion that GNU/Linux is a very decent choice for running your web / application servers. If they're better than Windows, we'll leave it up to you to decide. With the brilliant work done by organisations such as Red Hat and Canonical (Ubuntu's parent Organisation), Linux is being adopted on the desktop at a massive rate and not just in enterprise/scientific research environment, but by home users looking for safe and stable operating systems. This makes managing Linux-based systems an essential skill.

If you have multiple GNU/Linux-based systems on a common local network, you can use this guide to streamline their maintenance. You'll learn concepts such as centralisation and remote management among st others. It's important to understand that a Linux administrator is just a normal user with extended privileges in terms of file permission and is slightly more ninja with the command line than you are. Don't get intimidated by the term "Linux Administrator", if you've been using Linux for some time, you're already half way there.

This guide is designed as an introduction to basic Linux administration aimed especially towards home users and small system managers. Therefore, we have included tutorials on setting up media servers rather than having an LDAP authentication server.

There are appendices included with this guide that are independent of other chapters. We strongly recommended you to familiarise yourself with various terminologies by reading the appendices before other chapters. We have tried to focus on making this guide as applicable as possible, we have introduced you to setting a up and running an Apache web server, so that you can run a local web site for your place, or if your size escalates you can even have your own dedicated web server.

In order to utilise this guide to the maximum, it is strongly recommended to replicate the steps/commands mentioned here on a Linux system, a handy tip at this time would be that of virtual machines. Not only can you run a Virtual Machine of Linux if you are stuck in a windows environment because of an unknown force of the ancient Greeks, but also you can run multiple virtual machines on the same physical machine (assuming it can handle the load) to replicate a small network and experiment or play with it without causing any real damage. Consider it as your network simulator, so without any further delay, set one up and fire the terminal. d

1 Users' and User accounts

Linux has always pride itself in being a true multi-user operating system. Adding and removing new user accounts is one of the most common task for Linux administrators and it is imperative that he masters this simple and yet essential job. User accounts are also important from the security point of view, as a compromised account with sufficient privileges can make for an easy target.

The /etc/passwd file

Whenever you are attempting to logon to any Linux system, your identity is verified by the `/etc/passwd` file. Let us look at its content in a little more detail.

Each line in the file represents one user and contains seven fields separated by colons:

- Login name
- Encrypted password or password placeholder
- UID (user ID) number
- Default GID (group ID) number
- “GECOS” information: full name, office, extension, home phone
- Home directory
- Login shell

Most modern Linux systems don't show the encrypted password now, since with modern hardware they can be quickly decrypted compromising on the security. They are stored typically in a shadow file that isn't readable in text format. This mechanism of hiding even the encrypted password is known as the shadow password mechanism.

For example, the entry for the user `ntp` [Network time protocol] in the `/etc/passwd` file is as follows

```
ntp:x:116:124::/home/ntp:/bin/false
```

We will now look at each of the seven fields in a little more detail.

Login Name

Needs no explanation

Password

The `x` character is displayed when the user's password is stored in /etc/shadow in an encrypted format.

User Identification Number or UID

UIDs are 32-bit unsigned integers. It is a recommended practice though, to limit the largest value to a 16-bit range in order to maintain compatibility with older systems. However, if your requirements aren't met by the 16-bit range, feel free to use the whole spectrum.

The super user or `root`, will by definition always possess UID 0. You will also notice if you open the list of users on your system that there are user accounts created for deamons etc, ensure your manually set UID if any doesn't clash with them.

GID or Group Identification Number

This is very similar to the UID. We will just enumerate the reserved GIDs: `GID 0` is reserved for `root`, while `GID 1` is reserved for `bin` and the `daemon` group takes `GID 2`. The group listing and other details can be found in the `/etc/group` file.

GECOS field

Think of it as the address book detail field, there are no fixed formats for this, and varies from system to system. The next two fields will turn out to be of more importance.

Home directory

It can be an important consideraton to decide as to where to locate the home directory, especially in a networked environment, where your users might log on from multiple machines acting as thin clients, if your home directories are mounted over the network, and are unavailable in the event of a server crash.

Login shell

The login shell is normally a command interpreter such as the Bourne shell or the C shell (`/bin/sh` or `/bin/csh`), but it can be any program. Bash is the default and is used if `/etc/passwd` does not specify a login shell. Most users would be satisfied by this and if they want to make switch, you can always offer the alternatvie of the slightly more advanced Korn Shell, referred to as the `ksh`.

1.2 Adding and deleting users

The process for managing local users and groups is very straight forward and differs very little from most GNU/Linux operating systems. Ubuntu and other Debian based distributions, use "adduser" package for account management.

To add a user account, use the following command systax, and follow the prompt to give the account a password and other identifiable characteristics:

```
sudo adduser username
```

To delete a user account and its primary group, use the following command syntax:

```
sudo deluser username
```

Just deleting an account will not remove their respective home folder. It's up to you whether or not you wish to delete the folder manually or keep it according to your desired retention policies. Remember, any user added later on with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.

You may want to change these UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts:

```
sudo chown -R root:root /home/username/
sudo mkdir /home/archived_users/
sudo mv /home/username /home/archived_users/
```

To temporarily lock or unlock a user account, use the following syntax, respectively:

```
sudo passwd -l username
sudo passwd -u username
```

To add or delete a personalized group, use the following syntax, respectively:

```
sudo addgroup groupname
sudo delgroup groupname
```

To add a user to a group, use the following syntax:

```
sudo adduser username groupname
```

Where is root?

Ubuntu developers made a conscientious decision to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted or that it may not be accessed. It merely has been given a password that matches no possible encrypted value, and therefore, may not log in directly by itself.

Instead, users are encouraged to make use of a tool called `sudo` to carry out system administrative duties. `Sudo` allows an authorised user to temporarily elevate their privileges using their own password instead of having to know the password of the root account. This simple, yet, effective methodology provides accountability for all user actions, and gives the administrator granular control over which actions a user can perform with the said privileges.

If, for some reason, you wish to enable the root account, simply give it a password:

```
sudo passwd
```

`Sudo` will prompt you for your password, and then ask you to supply a new password for root as shown below:

```
[sudo] password for username: (enter your own password)
Enter new UNIX password: (enter a new password for root)
```

```
Retype new UNIX password: (repeat new password for root)
passwd: password updated successfully
```

To disable the root account, use the following passwd syntax:

```
sudo passwd -l root
```

You should read more on Sudo by checking out it's man page:

```
man sudo
```

By default, the initial user created by the Ubuntu installer is a member of the group `admin` that is added to the file `/etc/sudoers` as an authorised `sudo` user. If you wish to give any other account full root access through `sudo`, simply add them to the admin group.

User profile security

When a new user is created, the add user utility creates a brand new home directory named `/home/username`, respectively. The default profile is modelled after the contents found in the directory of `/etc/skel`, which includes all profile basics.

If your server will be home to multiple users, you should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read/ execute permissions. This means that all users can browse and access the contents of other users home directories. This may not be suitable for your environment.

To verify your current users home directory permissions, use the following syntax:

```
ls -ld /home/username
```

The following output shows that the directory /home/username has world readable permissions:

```
drwxr-xr-x  2 username username    4096 2010-06-12 20:03 username
```

You can remove the world readable permissions using the following syntax:

```
sudo chmod 0750 /home/username
```

Some people tend to use the recursive option (-R) indiscriminately which modifies all child folders and files, but this is not necessary, and may yield other undesirable results. The parent directory alone is sufficient for preventing unauthorized access to anything below the parent.

A much more efficient approach to the matter would be to modify the adduser global default permissions when creating user home folders. Simply edit the file /etc/adduser.conf and modify the DIR_MODE variable to something appropriate, so that all new home directories will receive the correct permissions.

```
   DIR_MODE=0750
```

After correcting the directory permissions using any of the previously mentioned techniques, verify the results using the following syntax:

```
   ls -ld /home/username
```

The results below show that world readable permissions have been removed:

```
   drwxr-x---    2 username username     4096 2010-06-12
20:04 username
```

Password policy

A strong password policy is one of the most important aspects of your security posture. Many successful security breaches involve simple brute force and dictionary attacks against weak passwords. If you intend to offer any form of remote access involving your local password system, make sure you adequately address minimum password complexity requirements, maximum password lifetimes, and frequent audits of your authentication systems.

Minimum password length

By default, Ubuntu requires a minimum password length of 4 characters, as well as some basic entropy checks. These values are controlled in the file /etc/pam.d/common-password, which is outlined below.

```
   password   required   pam_unix.so nullok obscure min=4
max=8 md5
```

If you would like to adjust the minimum length to 6 characters, change the appropriate variable to `min=6`. The modification is outlined below.

```
   password   required   pam_unix.so nullok obscure min=6
max=8 md5
```

Password expiration

When creating user accounts, you should make it a policy to have a minimum and maximum password age forcing users to change their passwords when they expire.

To easily view the current status of a user account, use the following syntax:

```
   sudo chage -l username
```

The output below shows interesting facts about the user account, namely that there are no policies applied:

```
   Last password change                                  :
Jan 20, 2010
   Password expires                                      :
never
```

```
  Password inactive                                         :
never
  Account expires                                           :
never
  Minimum number of days between password change            :
0
  Maximum number of days between password change            :
99999
  Number of days of warning before password expires         :
7
```

To set any of these values, simply use the following syntax, and follow the interactive prompts:

```
sudo chage username
```

The following is also an example of how you can manually change the explicit expiration date (`-E`) to `03/31/2010`, minimum password age (`-m`) of `5 days`, maximum password age (`-M`) of `90 days`, inactivity period (`-I`) of `5 days` after password expiration, and a warning time period (`-W`) of `14 days` before password expiration.

```
sudo chage -E 03/31/2010 -m 5 -M 90 -I 30 -W 14 username
```

To verify changes, use the same syntax as mentioned previously:

```
sudo chage -l username
```

The output below shows the new policies that have been established for the account:

```
  Last password change                              : Mar 20, 2010
  Password expires                                  : Jun 19, 2010
  Password inactive                                 : Aug 19, 2010
  Account expires                                   : Jan 31, 2010
  Minimum number of days between password change          : 5
  Maximum number of days between password change         : 90
  Number of days of warning before password expires      : 14
```

Other Security Considerations

Many applications use alternate authentication mechanisms that can be easily overlooked by even experienced system administrators. Therefore, it is important to understand and control how users authenticate and gain access to services and applications on your server.

SSH Access by Disabled Users

Simply disabling/locking a user account will not prevent a user from logging into your server remotely if they have previously set up RSA public key authentication. They will still be able to gain shell access to the server, without the need for any password. Remember to check the users home directory for files that will allow for this type of authenticated SSH access. e.g. `/home/username/.ssh/authorized_keys`.

Remove or rename the directory .ssh/ in the user's home folder to prevent further SSH authentication capabilities.

Be sure to check for any established SSH connections by the disabled user, as it is possible they may have existing inbound or outbound connections. Kill any that are found.

Restrict SSH access to only user accounts that should have it. For example, you may create a group called `sshlogin` and add the group name as the value associated with the AllowGroups variable located in the file `/etc/ssh/sshd_config`.

```
AllowGroups sshlogin
```

Then add your permitted SSH users to the group `sshlogin`, and restart the SSH service.

```
sudo adduser username sshlogin
sudo /etc/init.d/ssh restart
```

1.3 Creating a user account in Ubuntu with only web-surfing enabled

In many situation's, we come across a need to enable only a certain application for the guest user and we have to protect our files, directories and applications from their access. A good example is a browsing centre that provides the user just browsing facility.

What we're going to do...

Create a guest account and enable only the web browser application.

Step 1: Creating the unprivileged user account.
We can create the new unprivileged user account named guest here.
Go to `System > Administration > Users & Groups`.
Unlock the application by providing your password and click on the `Add User` button.
Fill the details like user name as guest (or anything as you wish, but don't forget to replace the following commands for the word guest), password, and select the profile as unprivileged user, and click `OK`.
Step 2: Configuring the user session.
Now we have to configure the user to run the web browser alone. For this we have to create the xsession configuration. Open the xsession file of the guest user by following command.

```
sudo gedit /home/guest/.xsession
And type the following into the file and save it.
/usr/bin/metacity &
/usr/bin/firefox
```

Metacity is the a window manager used by default in the GNOME Desktop Environment, and Firefox is the web browser of choice. People can change this to run any application they need.

Now run the following command to change the permissions of the file

```
sudo chmod +x /home/guest/.xsession
```

Now you can set up your guest acount as a automatic login or a timed login using Advanced Tab in

```
System > Administration > Login Window
```

From now, when we log into the guest account, the Firefox browser alone will be opened. The user cannot run any other applications or cannot navigate to any other file system.

Once you minimise the window, you will find only the black screen and the only way to bring up the window back is `[Alt] + [Tab]`.

If we close the window, we will be logged out. Enjoy the privacy and security that Linux offers.

It is also possible to add new users from the command line. To do so, start a terminal window session and at the command prompt enter a command similar to:

```
sudo adduser --home /home/ck jck
```

The above command will prompt for a password for the account and optional contact information. Once the information has been gathered, `adduser` creates the new account and the `/home/jck  home` directory. The `adduser` tool provides a number of different options, details of which can be learned by reviewing the adduser man page as follows:

```
man adduser
```

1.4 Deleting a user from an Ubuntu Linux system

An existing user may be deleted using the same user settings dialog used to add a user as outlined above. Select the System desktop menu and choose Users and Groups from the Administration sub-menu to launch the User settings dialog.

Select the user to be deleted and click on Delete. A confirmation dialog will appear. If you wish to proceed click on Delete in the confirmation dialog to commit the change.

Note that although the deletion process will remove the account, it will leave the user's home directory intact. This will need to be deleted manually, if this information and any others files therein, are no longer required.

A user account may also be deleted from command-line using the deluser utility:

```
sudo deluser jck
```

It is also possible to remove the user's home directory as part of the deletion process:

```
sudo deluser --remove-home jck
```

Alternatively, all files owned by the user, including those in the user's home directory may be removed as follows:

```
sudo deluser --remove-all-files jck
```

The files in the user's home directory can also be backed up to another location before the directory is deleted using `--backup-to command-line` option together with the path to the backup directory:

```
sudo deluser --backup-to /oldusers/backups/jck --remove-
home jck
```

1.5 Modifying an Ubuntu Linux Group

To add a group from the command line, use the `addgroup` utility. For example:

```
sudo addgroup accounts
To add an existing user to an existing group:
sudo adduser jck accounts
```

1.6 Deleting a Group from an Ubuntu Linux System

A group may be deleted from a system using the delgroup utility:

```
sudo delgroup accounts
```

Note that if the group to be deleted is the primary group for any user it cannot be deleted. A group can be deleted only if it is empty using the following command:

```
sudo delgroup --only-if-empty accounts
```

To remove a user from membership of a group use the following command syntax:

```
sudo deluser jck accounts
```

2 Hardware device files

In Linux all the hardware devices are treated as files. They allow transparent communication between user space applications and hardware devices. A device file is an interface for a device driver that appears as if it were an ordinary file. Device files provide simple interfaces to peripheral devices such as printers in addition to allowing access to specific resources on devices such as hard disk partitions. In addition to providing interfaces to hardware devices, device files provide access to system resources that have no actual connection with any actual device such as random number generators or data sinks. Device files allow users or software to interact with devive drivers using system calls or standard input/output. This simplifies several tasks.

There are two types of device files, character devices and block devices:

Character devices communicate to devices through which the system transmits data one character at a time. Examples of such devices are serial modems, virtual terminals, mice and keyboards. Such devices usually do not support random access to data.

Block devices relate to devices through which data is moved in the form of blocks (one block can range from 512 bytes to 32 kilobytes). Examples of such devices are hard disks, CD or DVD ROMs and memory regions.

All the device files are located in the `/dev` directory. You can find out the type of a particular device file by using `ls -l` or `file`. If you use `ls -l` the first letter of the permission string shows the device file's type. If it is c, then it means that it is a character special file and if it is b it means that it is a block special file. You can also use `cat /proc/devices` to get a list of all character and block files.

Example:

```
$ ls -l /dev/tty1
crw------- 1 jck tty 4, 1 Jun 15 21:08 /dev/tty1
$ file /dev/tty1
/dev/tty1: character special

$ ls -l /dev/sda1
brw-rw---- 1 root disk 8, 1 Jun 15 18:08 /dev/sda1
$ file /dev/sda1
/dev/sda1: block special
```

2.1 Naming conventions

These prefixes are commonly used in Linux systems to identify device drivers in the `/dev` hierarchy:

1. fb: frame buffer (video output device)

fd: (platform) floppy disks, though this same abbreviation is also commonly used to refer to file descriptor
hd: ("classic") IDE driver
hda: the master device on the first ATA channel
hdb: the slave device on the first ATA channel
hdc: the master device on the second ATA channel
hdd: the slave device on the second ATA channel
lp: line printers
parport, pp: parallel ports
pt: pseudo-terminals (virtual terminals)
SCSI driver, also used by libata (modern PATA/SATA driver), USB, Firewire, etc.
sd: mass-storage driver
sda: first registered device
ses: Enclosure driver
sg: generic SCSI layer
sr: "ROM" driver (data-oriented optical disc drives; scd is just a secondary alias)
st: magnetic tape driver
tty: terminals
ttyS: (platform) serial port driver
ttyUSB: USB serial converters, modems, etc.

This prefix is usually followed by a number which uniquely identifies the particular device. For hard drices an alphabet is used to identify the device and it is followed by a number to identify partitions. For example, /dev/sda corresponds to the entire disk and /dev/sda1,/dev/sda2 correspond to the first and second partitions respectively.

2.2 Pseudo devices

In addition to corresponding to physical devices, device nodes also correspond to pseudo-devices that provides various functions handled by the operating system. Some of the most commonly used pseudo devices include:

`/dev/null`
`/dev/null` accepts all imputs and discards it. It produces no output. It is commonly used when the user has no use for the standard output and the program has no option to supress standard output.

`/dev/zero`
This produces a continuous stream of zero value bytes.

/dev/random
This produces a variable length stream of random numbers.

2.3 Optimising the boot process

If you want to optimise your boot process, your first step should be to analyse how the time is being spent while your computer boots and then optimise the slow bits. You can visualise your boot process by using a tool called `bootchart`. Install it using synaptic or `apt-get`:

```
$ sudo apt-get install bootchart
```

Now reboot your machine and you will find your bootchart in `/var/log/bootchart` as a png file. Remember that bootchart will run in every boot henceforth, so don't forget to disable or uninstall it when you no longer need it. You can uninstall it using apt-get:

```
$ sudo apt-get remove bootchart
```

If you dont want to uninstall it, remove its SysV script from executing after startup:

```
$ cd /etc/init.d
$ sudo update-rc.d -f stop-bootchart remove
```

If you want to re-enable bootchart you may either reinstall it trough the repositories or add it back to runlevels 2, 3, 4, 5:

```
$ cd /etc/init.d
$ sudo update-rc.d stop-bootchart start 99 2 3 4 5 .
```

Now that you have your boot process charted out it is time to start optimising your boot process.

Firstly you can reduce the time grub waits before it loads the kernel, the default is 5 seconds.

Now examine your bootchart and stop the programs which take too long to boot which you don't need from loading during boot. You can remove them by going to `System > Preferences > Startup Applications` and unchecking the things you don't need.

ureadahead is a process that pulls files into memory that it knows it is going to need during the boot process to make it faster. If you notice that ureadahead spends a lot of time on your bootchart it is possible that the ureadahead pack files are wrong or corrupt and that is causing it to do odd things. If this is the case you need to regenerate the ureadahead pack files , you can do so like this:

```
$ sudo rm /var/lib/ureadahead/*pack
```

The next time you reboot ureadahead will regenerate those files and you can reboot again to see if that helped.

You can also profile your boot to achieve similar results:

Profile your boot::

If you ask the kernel to profike your boot it will determine what files are accessed and then sort them according to how they are stored on your hard disk. The boot time during profiling will be long but your subsequent boots should be considerably faster. Follow these steps to profile your boot:

1.At your boot screen when grub loads press `[E]` (edit).
2.Navigate to the entry beginning with "kernel" and press `[E]` again.
3.Add `profile` without quotes at the end of this line and hit `[Enter]`.
4.Press `[B]` for booting.

If you want to eleminate rebooting after updating the kernel on your server you can install kexec for warm reboots.

If you do a warm reboot you can avoid waiting while your computer re-initialises its hardware and you can skip going through the bios and bootloader. Your system can simply go to a minimal runlevel and load the new kernel image into memory and come back up. Make sure not to let kexec set itself up as the default restart handler if your system is a dual boot system and you want to reboot into a different O.S often. Follow these steps to configure kexec:

First install kexec

```
$ sudo apt-get install kexec-tools
```

Then make a backup of the reboot script and edit it

```
$ sudo cp /etc/init.d/reboot{,.bkp};sudo gedit /etc/
init.d/reboot
```

Find the do_stop function which looks like this:

```
  do_stop                    ()                {
…                            .                 .
…                            .                 .
log_action_msg    "Will      now     restart"
reboot            -d         -f                -i
}
  And replace it with this:
  do_stop () {
        ….
        ….
        log_action_msg "Will now restart"
        if [ -x /sbin/kexec ]; then
                  kexec -l --append="`cat /proc/cmdline`"
--initrd=/boot/initrd.img-`ls /lib/modules | sort -nr |
head -n 1` /boot/vmlinuz-`ls /lib/modules | sort -nr |
head -n 1`
               sync
```

```
                umount -a
                kexec -e
            else
                reboot -d -f -i
            fi
  }
```

You can even use sysv-rc-conf (`sudo apt-get install sysv-rc-conf`) to change the enable/disable settings. Some of the services to consider disabling include:

`anacron`
As mentioned earlier, this subsystem periodically runs processes. You may want to disable it and move any critical services to cron.

`atd and cron`
By default, there are not at or cron jobs scheduled. If you do not need these services, then they can be disabled. Personally, I would always leave them enabled since they take relatively few resources.

`apmd`
This service handles power management and is intended for older systems that do not support the ACPI interface. It only monitors the battery. If you have a newer laptop (or are not using a laptop), then you probably do not need this service enabled.

`acpid`
The acpid service monitors battery levels and special laptop buttons such as screen brightness, volume control, and wireless on/off. Although intended for laptops, it can also support some desktop computers that use special keys on the keyboard (for example, a www button to start the browser). If you are not using a laptop and do not have special buttons on your keyboard, then you probably do not need this service.

`bluez-utiles`
This provides support for Bluetooth devices. If you don't have any, then this can be disabled.

`dns-clean, ppp, and pppd-dns`
These services are used for dynamic, dial-up connections. If you do not use dialup, then these can be disabled.

`hdparm`
This system is used to tune disk drive performance. It is not essential and, unless configured, does not do anything. The configuration file is /etc/hdparm.conf and it is not enabled by default.

`hplip`
This provides Linux support for the HP Linux Image and Printing system. If you do not need it, then it can be disabled. Without this, you can still print using the lpr and CUPS systems.

`mdadm, mdadm-raid, and lvm`
These provide file system support for RAID (mdadm and mdadm-raid) and Logical Volume groups (lvm). If you do not use either, then these can be disabled.

`nfs-common, nfs-kernel-server, and portmap`
These are used by NFS-they are only present if you installed NFS support. If you do not need NFS all the time, then you can disable these and only start the services when you need them:

```
sudo /etc/init.d/portmap start
sudo /etc/init.d/nfs-common start
sudo /etc/init.d/nfs-kernel-server start
```

`pcmcia and pcmciautils`
These provide support for PCMCIA devices on laptops. If you do not have any PCMCIA slots on your computer, then you do not need these services.

`powernowd and powernowd.early`
These services are used to control variable-speed CPUs. Newer computers and laptops should have these enabled, but older systems (for example, my dual-processor 200 MHz PC) do not need it.

`readahead and readahead-desktop`
These services are used to preload libraries so some applications will initially start faster. In a tradeoff for speed, these services slow down the initial boot time of the system and consume virtual memory with preloaded libraries. If you have limited RAM, then you should consider disabling these services.

`rsync`
This is a replacement for the remote copy (`rcp`) command. Few people need this-it is used to synchronize files between computers.

`vbesave`
This services monitors the Video BIOS real-time configuration. This is an ACPI function and is usually used on laptops when switching between the laptop display and an external display. If your computer does not support APCI or does not switch between displays, then you do not need this service.

Although there are many services that you probably do not need, there are a few that are essential. You should not turn off these essential services unless you really know what you are doing:

dbus
Provides messaging services.

`gdm`
This is the Gnome Desktop. Only disable this if you do not want a graphical desktop.

`klogd`
This is the kernel log daemon. Removing it disables system logging.

`makedev and udev`
These create all device nodes.

`module-init-tools`
Loads kernel modules specified in `/etc/modules`.

`networking and loopback`
These start and stop the network. Disabling removes the network configuration at boot.

`procps.sh`
Any kernel tuning parameters added to /etc/sysctl.conf are processed by this service.

`urandom`
This seeds the real random number generator that is used by most cryptographic system. You should leave it enabled.

As a rule of thumb, if you do not know what it is, then leave it on. Also, if the service only runs in single-user mode (rcS) that it is usually smart to not change it. Single user mode is where you should go when everything fails in order to repair the system

3 Directory structure and filesystem

3.1 Directory structure

Let us get familiar with the directory structure of your Ubuntu set-up. The file system is typical Unix, with a hierarchal structure, best understood as a tree-like structure for those of you familiar with algorithmic design. The top of the tree, i.e., the highest level in the file system is the root directory, or / and everything, including hard-disks, partitions, USB drives, directories and data files fall below the / as individual files. For windows users, the important thing to understand here is that unlike Windows, even hard disks and removable media devices are treated as files, not some special category, so you get more power in configuring all your devices.

The root directory of Ubuntu contains some important system directories, which are common with most other Linux distributions:

- `bin` – This directory contains important binary applications
- `boot` - This directory contains the files needed for booting up the operating system
- `dev` - This directory contains the hardware device files that we will talk in detail in a later section
- `etc` – This is the place where you can find the configuration files of the base OS and other installed applications and startup scripts
- `home` – This is your home directory where all your personal data is stored
- `lib` – The system libraries required for the proper functioning of installed software are present in this directory
- `lost+found` – As the name suggests, this contains the lost and found files of your / directory.
- `media` – This directory is where all your mounted (loaded) removable media such as CDs and digital cameras will figure up as files, for you to configure
- `mnt` – This directory contains the mounted filesystems of your computer
- `opt` – This directory provides a location for optional applications to be installed
- `proc` – It's a special dynamic directory that maintains information about the state of the system, including currently running processes and their thread specific details
- `root` – This is commonly known as the slash-root directory, because we refer to / as root in Ubuntu. It's nothing, but the root user's home directory
- `sbin` – Some important system binaries are stored here
- `srv` – This directory acts as a temporary location for data meant to be used by servers
- `sys` – This directory contains system-specific information meant as

reference for other applications
- `tmp` – As the name suggests, it acts as storage for temporary files
- `usr` – This is where most of your applications and files will be stored, as anything present here is available for all users to access
- `var` – This is a directory for variable files such as logs and databases. Notice the contrast with the `/tmp` directory.

3.2 Converting an ext3 file system into ext4

Ext4 file system provides better performance and faster file system check than the ext3 file system which was in use until very recently. With Kernel 2.6.28, ext4 was marked stable, and with a subsequent Ubuntu 9.04 (Jaunty Jackalope) release, you can even use Ext4 for a fresh install. But since most of you are stuck with the older ext3 file system, we will see how to convert it into the new ext4 file system.

First thing that you should do is check your existing file system to be sure that you have an Ext4 aware kernel, else you might end up with a non-bootable brick. Type in the command

`uname -a` and make sure that its higher than `2.6.28`.

Next, you need to switch over to the Ext4 driver without changing the existing files on disk. This is made possible because of the fact that the Ext4 driver is backwards compatible and can mount an Ext3 file system. Type the following command:

```
sudo nano /etc/fstab
```

and look for the ext3 on the line that defines your disk(sda1 in most cases) and change it to `ext4`. If you have more than one partition for your Linux set-up, change for all of them and reboot your computer.

After this, you are ready to enable the new Ext4 features for your system with the following command:

```
sudo tune2fs -O extents,uninit_bg,dir_index <dev>
```

where dev stands for the disk edited in the previous step, `/dev/sda1` in our example. You need to again reboot your system for this to take effect and to run a file system check which can only be done on an unmounted file system.

You might see a lot of warnings of file system issues on your screen. It is perfectly normal and you've nothing to worry about. All that is left now is reinstalling GRUB. The GRUB boot loader that you will install will also, obviously have to be a version later than when ext4 was marked stable, so that it understands your file system and actually boots up your computer. Do a `sudo grub-install <dev>` to make sure that your grub version is the latest.

All the files written onto the disk after this will be able to take full advantage of the ext4, while your previous files will still be of ext3 format.

In a real world scenario, this converts into a decreased boost in speed after installing ext4, but as you continue using your computer, you will notice a gradual speed-up.

4 File permissions in Linux

4.1 Permissions and Restrictions

In the section on managing files and folders, we talked about how everything in Linux, be it a directory, an actual file, or even a device are just files to the operating system and all of these have permissions that allow or restrict others from viewing, modifying or executing them. If the file is of the type Directory then it has different permissions and restrictions than say, files and device nodes and each of these have access restrictions with permissions and user restrictions associated with owner/group association. However, the super user `root` has the ability to access any file on the system. These permissions are referred to as bits. If the owner read & execute bits are on, then the corresponding permissions are `-r-x------`

There are three types of access restrictions:

Permission	Action	chmod option
read	(view)	r or 4
write	(edit)	w or 2
execute	(execute)	x or 1

There are also three types of user restrictions:

User	ls output
owner	-rwx------
group	-rwx------
other	-rwx------

These restrictions, however are not inheritable, as in the case where restrictions are set for the owner's group or 'everyone', the file owner will still be unaffected with full rights.

We mentioned previously that directories have directory permissions, which are different from file and device node permissions. These are:

Permission	Action	chmod option
read	restricts or allows viewing the directories contents, i.e. ls command	r or 4
write	restricts or allows creating new files or deleting files in the directory	w or 2
execute	restricts or allows changing into the directory, i.e. cd command	x or 1

Example:

If a file is owned by the user root and belongs to the root group, the permissions for that file will be:

```
-rw-r--r--
```

owner Read & Write (`rw-`)
group Read (`r--`)
others Read (`r--`)

This is very similar to the permissions that your `/etc/shadow` file (which contains all local user passwords) will have:

```
user@host:/home/user# ls -l /etc/shadow
-rw-r-----   1 root shadow 869 2010-06-14 18:27 /etc/shadow
user@host:/home/user#
```

Permissions:
owner = Read & Write (`rw-`)
group = Read (`r--`)
other = None (`---`)
Ownership:
owner = root
group = shadow

Now that you are familiar with permissions and understand what the `ls` output means, let us get on with changing permissions. The command that we will be using here is chmod. A very important thing here is to be cautious while playing around with permissions, because you do not want to create security flaws on your system, or give unrestricted access to unauthorised personnel.

Chmod can be used with letters, as well as numbers, which should be pretty obvious to you if you understood the permission tables given above. While using the letters, the various options in the `chmod {options} filename` command that you have are in the accompanying table.

Options	Definition
u	owner
g	group
o	other
x	execute
w	write
r	read
+	add permission
-	remove permission
=	set permission

Example:

We have 4 files, named A, B, C and D, with read and write permission for the owner and read for the rest.

Now if we want to add owner execute bit to A, the command will be:

```
user@host:/home/user$ chmod u+x A
user@host:/home/user$ ls -l A
-rwxr--r--  1 user user 0 Jun 14 16:14 A
```

If we want to add other write & execute bit to B, the command will be:

```
user@host:/home/user$ chmod o+wx B
user@host:/home/user$ ls -l B
-rw-r--rwx  1 user user 0 Jun 14 16:15 B
```

If we want to remove group read bit for C, the command will be:

```
user@host:/home/user$ chmod g-r C
user@host:/home/user$ ls -l C
-rw----r--  1 user user 0 Jun 14 16:15 C
```

If we want to add read, write and execute bit to everyone for D, the command will be:

```
user@host:/home/user$ chmod ugo+rwx D
user@host:/home/user$ ls -l D
-rwxrwxrwx  1 user user 0 Jun 14 16:16 D
user@host:/home/user$
```

These examples should make it very clear to you as to how you can mix and match and set the desired permissions to all your files.

To use chmod with numbers, the following options can be used:

Options	Definition
#--	owner
-#-	group
--#	other
1	execute
2	write
4	reae

Owner, Group and Other are represented by three numbers. First, we determine the type of access needed for the file and then get the value for the options before adding.

Example: If you want a file to have `-rw-rw-rwx` permissions, you will use the following options:

```
user@host:/home/user$ chmod 667 XYZfile
```

If however, you want a file to have `--w-r-x--x` permissions, you will use the following:

```
user@host:/home/user$ chmod 251 ABCfile
```

Again taking the same route on files A, B, C and D, with read and write permission for the owner and read for the rest.

If we want to add owner execute bit to A, the command will be:

```
user@host:/home/user$ chmod 744 A
user@host:/home/user$ ls -l A
```

```
-rwxr--r--  1 user user 0 Jun 14 16:19 A
```

If we want to add other write & execute bit to B, the command will be:
user@host:/home/user$ chmod 647 B
user@host:/home/user$ ls -l B
-rw-r--rwx 1 user user 0 Jun 14 16:20 B

Owner	Group	Other
write	read & execute	execute
2	4+1=5	1

If we want to remove group read bit for C, the command will be:

```
user@host:/home/user$ chmod 604 C
user@host:/home/user$ ls -l C
-rw----r--  1 user user 0 Jun 14 16:24 C
```

If we want to add read, write and execute bit to everyone for D, the command will be:

```
user@host:/home/user$ chmod 777 D
user@host:/home/user$ ls -l D
-rwxrwxrwx  1 user user 0 Jun 14 16:24 D
user@host:/home/user$
```

You can even change permissions on files that you do not have the ownership of using the `sudo` command, if you do have the root password. Be careful while using sudo, else you might mess up your system a great deal.

Example:

```
user@host:/home/user$ ls -l /usr/local/bin/XYZfile
-rw-r--r--  1 root root 550 2010-06-13 21:53 /usr/local/
bin/XYZfile
user@host:/home/user$

user@host:/home/user$  sudo  chmod  o+x  /usr/local/bin/
XYZfile

user@host:/home/user$ ls -l /usr/local/bin/XYZfile
-rw-r--r-x  1 root root 550 2010-06-13 21:54 /usr/local/
bin/XYZfile
user@host:/home/user$
```

Now, if you want to change the permissions of multiple files and directories, you can do so using the `–R` option which basically means Recursive Permission change.

For example, if you intend to change the all the permissions of each file and folder under a specified directory at once, you can use the `sudo chmod` with `-R`:

```
  user@host:/home/user$  sudo  chmod  777  -R  /path/to/
PQRDirectory
  user@host:/home/user$ ls -l
  total 3
  -rwxrwxrwx  1 user user 0 Jun 14 11:45 Afile
  drwxrwxrwx  2 user user 4096 Jun 14 11:45 Xfolder
  -rwxrwxrwx  1 user user 0 Jun 14 11:45 Bfile
```

If you want to assign reasonably secure permissions to files and folders/ directories, you should give files a permission of `644`, and directories a `755` permission, since chmod `-R` assigns to both using the sudo command, the find command, and a pipemill to chmod as in the following example:

If you only want to change permission of files under a specified directory-

```
  user@host:/home/user$  sudo  find  /path/to/XYZDirectory
-type f -print0 | xargs -0 sudo chmod 644
  user@host:/home/user$ ls -l
  total 3
  -rw-r--r--  1 user user 0 Jun 14 11:48 13 A
  drwxrwxrwx  2 user user 4096 Jun 14 11:48 folderX
  -rw-r--r--  1 user user 0 Jun 14 11:48 B
```

If you want to only change permission of directories under a specified directory (including that directory):

```
  user@host:/home/user$  sudo  find  /path/to/XYZDirectory
-type d -print0 | xargs -0 sudo chmod 755
  user@host:/home/user$ ls -l
  total 3
  -rw-r--r--  1 user user 0 Jun 14 11:52 A
  drwxr--r--  2 user user 4096 Jun 14 11:52 folderX
  -rw-r--r--  1 user user 0 Jun 14 11:52 B
```

Besides giving you the options to change file permissions, Linux also gives you the option of changing the File Owner and Group using the command. For example, if you want to change the foobar file's owner to `PQR`, you need to type the following command:

```
  user@host:/home/user$ sudo chown PQR foobar
```

If you want to change the foobar file's group to XYZ, you can use either chgrp or chown with this syntax:

```
  user@host:/home/user$ sudo chgrp XYZ foobar
  user@host:/home/user$ sudo chown :XYZ foobar
```

And, if you want to change the foobar file's owner to PQR and the group to XYZ with a single command, you should use the following command:

```
  user@host:/home/user$ sudo chown PQR:XYZ foobar
```

4.2 Posix ACLs

If you're not satisfied with the standard UNIX file permissions, you can use the Posix ACLs from the acl package to achieve a finer granularity of permissions. To enable Posix ACLs, you need to install the acl package by the following command:

```
sudo apt-get install acl
```

Once that's done, you can use the Eiciel package from the repository that grants you GUI access to ACLs through the Nautilus file manager.

4.3 File removal

If you want to remove a file you cannot delete, you can use the following command:

```
  sudo rm -rf filename
```

4.4 Sticky Bit

The Sticky bit prevent users from altering or replacing any other user's files. So, it is advisable that all public directories be configured with sticky bit using the command:

```
chmod u+t <directory>
```

The `u` here adds the sticky bit to the user; `g` adds it to the group; and `o` adds it for others. The + means that you are adding the sticky bit. If you want to later remove it, you can replace it with a –.

5 Introduction to fstab

All the necessary information required to automate the process of mounting partions, i.e., the the process where a raw (physical) partition is prepared for access and assigned a location on the file system tree (or mount point) is contained in the configuration file /etc/fstab.
Partitions listed in fstab can be configured to automatically mount during the boot process. If a device/partition is not listed in fstab ONLY ROOT can mount the device/partition, whereas users can mount a device/partition if the device is in fstab with the proper options.

fields	description
<device>	The device/partition (by /dev location or UUID) that contain a file system.
<mount point>	The directory on your root file system (aka mount point) from which it will be possible to access the content of the device/partition (note: swap has no mount point). Mount points should not have spaces in the names.
<file system type>	Type of file system
<options>	Mount options of access to the device/partition (see the man page for mount).
<dump>	Enable or disable backing up of the device/partition (the command dump). This field is usually set to 0, which disables it.
<pass num>	Controls the order in which fsck checks the device/partition for errors at boot time. The root device should be 1. Other partitions should be 2, or 0 to disable checking.

5.1 Fstab file configuration

The syntax of a fstab entry is :

```
 [Device]  [Mount  Point]  [File  System  Type]  [Options]
[Dump]  [Pass]
```

Device

By default, Ubuntu now uses UUID to identify partitions.

UUID=xxx.yyy.zzz

To list your devices by UUID use blkid

sudo blkid

Alternately syntax to refer to partitions :

- Device : /dev/sdxy
- Label : LABEL=label
- Network ID
- `Samba : //server/share`
- `NFS : server:/share`

- `SSHFS : sshfs#user@server:/share`

5.2 Mount point

A mount point is a location on your directory tree to mount the partition. The default location is /media although you may use alternate locations such as /mnt or your home directory.

You may use any name you wish for the mount point, but you must create the mount point before you mount the partition.

For example : /media/windows
sudo mkdir /media/windows

5.3 File System Type

You may either use auto or specify a file system. Auto will attempt to automatically detect the file system of the target file system and in general works well. In general auto is used for removable devices and a specific file system or network protocol for network shares.

Examples:

- auto
- vfat - used for FAT partitions.
- ntfs, ntfs-3g - used for ntfs partitions.
- ext2, ext3, jfs, reiserfs, etc.
- udf,iso9660 - for CD/DVD.
- swap.

5.4 Options

Options are dependent on the file system.
You may use "defaults" here and some typical options may include :
• defaults = rw, suid, dev, exec, auto, nouser, and async.
• /home = The options for a separate home partition should be nodev,nosuid
• ntfs/vfat = permissions are set at the time of mounting the partition with umask, dmask, and fmask and cannot be changed with commands such as `chown` or `chmod`.

We advise `dmask=027,fmask=137` (if you use `umask=000` all your files will be executable). More permissive options would be `dmask=000`, `fmask=111`.

For mounting samba shares you can specify a username and password, or better a credentials file. The credentials file contains should be owned by root.root with permissions = 0400 .

5.5 Common options

• sync/async - All I/O to the file system should be done (a)synchronously.
• auto - The filesystem can be mounted automatically (at bootup, or when

mount is passed the -a option). This is really unnecessary as this is the default action of mount -a anyway.

• `noauto` - The filesystem will NOT be automatically mounted at startup, or when mount passed -a. You must explicitly mount the filesystem.

• `dev/nodev` - Interpret/Do not interpret character or block special devices on the file system.

• `exec / noexec` - Permit/Prevent the execution of binaries from the filesystem.

• `suid/nosuid` - Permit/Block the operation of suid, and sgid bits.

• `ro` - Mount read-only.

• `rw` - Mount read-write.

• `user` - Permit any user to mount the filesystem. This automatically implies noexec, nosuid,nodev unless overridden.

• `nouser` - Only permit root to mount the filesystem. This is also a default setting.

• `defaults` - Use default settings. Equivalent to rw, suid, dev, exec, auto, nouser, async.

• `_netdev` - this is a network device, mount it after bringing up the network. Only valid with fstype nfs.

5.6 Dump

This field sets whether the backup utility dump will backup file system. If set to "0" file system ignored, "1" file system is backed up. Dump is seldom used and if in doubt use 0.

5.7 Pass (fsck order)

Fsck order is to tell fsck what order to check the file systems, if set to "0" file system is ignored. Often a source of confusion, there are only 3 options :

• 0 == do not check.

• 1 == check this partition first.

• 2 == check this partition(s) next

In practice, use "1" for your root partition, / and 2 for the rest. All partitions marked with a "2" are checked in sequence and you do not need to specify an order.

Use "0" to disable checking the file system at boot or for network shares.

You may also "tune" or set the frequency of file checks (default is every 30 mounts) but in general these checks are designed to maintain the integrity of your file system and thus you should strongly consider keeping the default settings.

5.8 File system specific examples

Here are a couple of basic examples for different file system types. I will use

/dev/sdb1 or /dev/hda2 for simplicity, but remember that any /dev location, UUID=<some_id>, or LABEL=<some_label> can work.

ext2 and ext3

The main difference between ext2 and ext3 is that ext3 has journaling which helps protect it from errors when the system crashes.

A root filesystem:

UUID=30fcb748-ad1e-4228-af2f-951e8e7b56df / ext3 defaults,errors=remount-ro,noatime 0 1

A non-root file system, ext2:

/dev/sdb1 /media/disk2 ext2 defaults 0 2

fat16 and fat32

/dev/hda2 /media/data1 vfat defaults,user,exec,uid=1000,gid=100,umask=000 0 0

/dev/sdb1 /media/data2 vfat defaults,user,dmask=027,fmask=137 0 0

ntfs

This example is perfect for a Windows partition.

/dev/hda2 /media/windows ntfs-3g defaults,locale=en_US.utf8 0 0

For a list of locales available on your system, run

- locale -a

hfs+

the hfs+ filesystem is generally used by Apple computers.

/dev/sdb1 /media/Macintosh_HD hfsplus rw,exec,auto,users 0 0

5.9 Editing fstab

Please, before you edit system files, make a backup. The -B flag with nano will make a backup automatically.

To edit the file in Ubuntu, run:

gksu gedit /etc/fstab

To edit the file in Kubuntu, run:

kdesu kate /etc/fstab

To edit the file directly in terminal, run:

sudo nano -Bw /etc/fstab

- -B = Backup origional fstab to /etc/fstab~ .
- -w = disable wrap of long lines.

Alternate:

```
sudo -e /etc/fstab
```

5.10 How to label

How the label and the UUID are set depends on the file system type used. It can normally be set when creating/formatting the file system and the file system type usually has some tool to change it later on (e.g. e2tunefs,xfs_admin,reiserfstune,etc.)

```
Mke2fs/e2label/tune2fs
```

Note: For either ext2 or ext3 file systems.

WARNING: mke2fs will reformat your partition and set a label at the same time. This will delete any data on the target partition.

To set a label without reformatting use e2label or tune2fs

• Make a label:

```
mke2fs -L <label> <dev>
```

OR

```
e2label <dev> <label>
```

OR

```
tune2fs -L <label> <dev>
```

Examples:

```
mke2fs -L data /dev/hda3
```

OR

```
e2label /dev/hda3 data
```

OR

```
tune2fs -L data /dev/hda3
```

• Create a mount point:

```
sudo mkdir /media/data
```

• Add an entry to /etc/fstab:

```
LABEL=data /media/data ext3 defaults 0 0
```

• To mount:

```
sudo mount LABEL=data
```

```
ReiserFS
reiserfstune --l <Label> <device>
```

Note

That is a small "L" and not the number 1.

```
JFS
jfs_tune -L <Label> <device>
```

To show the label:

```
jfs_tune -l <device>
```

Note

That is a small "L" and not the number 1.

```
XFS
```

sudo xfs_admin -L <Label> <device>

To show the label:

```
xfs_admin -l <device>
```

Note
That is a small "L" and not the number 1.

FAT (Windows partitions)

Use mtools to label a FAT partition:

• Install mtools:

sudo aptitude install mtools

• Copy the mtools configuration file to ~:

cp /etc/mtools.conf ~/.mtoolsrc

Note: ~ is shorthand for /home/user_name.

• Mount your flash drive.

• Edit ~/.mtoolsrc:

gedit ~/.mtoolsrc

• Add these lines to the end of ~/.mtoolsrc:

drive i: file="<device>"

mtools_skip_check=1

Where <device> is the device assigned to your mounted USB device/flash drive (ie sda1, sdb1, ...).

Note
You can do this from the command line:

```
echo 'drive i: file="<device>"' >> ~/.mtoolsrc
echo mtools_skip_check=1 >> ~/.mtoolsrc
```

Although you will need to edit ~/.mtoolsrc for each new device if the device assignment changes.

Example: = `drive i: file="/dev/sda1"`

• Change to drive i:

```
mcd i:
```

• Check the current label:

```
mlabel -s i:
```

• Change the current label:

```
sudo mlabel -s i:DATA
```

Or

```
sudo mlabel i:DATA
```

pieroxy reports the -s flag did not work, thanks pieroxy

Note
mlabel USES ALL CAPS.

• Add an entry to fstab:

```
LABEL=DATA <mount_point> vfat defaults 0 0
```

Note

You can also mount the usb device with:

```
mount LABEL=<label>
```

NTFS (Windows partitions):

First install ntfsprogs:

sudo aptitude install ntfsprogs

• Show label:

```
ntfslabel <device>
```

• Change label:

```
ntfslabel <device> <label>
```

Where:

`<label>` = your new label

`<device>` = your partition to label (/dev/hda1 perhaps)

• Add an entry to fstab:

```
LABEL=DATA <mount_point> ntfs(or ntfs-3g) defaults 0 0
```

Note

You can also mount the usb device with:

```
mount LABEL=<label>
```

Useful Commands

To view the contents of /etc/fstab, run the following terminal command:

```
cat /etc/fstab
```

To get a list of all the UUIDs, use one of the following two commands:

```
sudo blkid
ls -l /dev/disk/by-uuid
```

To list the drives and relevant partitions that are attached to your system, run:

```
sudo fdisk -l
```

To mount all file systems in /etc/fstab, run:

```
sudo mount -a
```

Remember that the mount point must already exist, otherwise the entry will not mount on the filesystem. To create a new mount point, use root privileges to create the mount point. Here is the generalization and an example:

```
sudo mkdir /path/to/mountpoint
sudo mkdir /media/disk2
```

6 Setting up a local APT repository

If you have a plethora of machines running on a local server, which you probably have, since you are reading this guide, you will notice that it is a wasteful process of downloading packages from each machine from the internet, of course you have the alternative for syncing the `/var/apt/cache` for each machine, but that is a tedious and cumbersome process, and would not work for upgrades, instead the recommended way is to set up a local repository via the `apt-cacher`. This way, you won't download common packages more than once from official repositories. Here is the situation, we have one machine called repository-cache, this machine is going to act as the repository cache, basically, any other machines in your network is going to use it as a repository.

On your terminal, type:

```
$ sudo apt-get install apt-cacher
```

Now, we need to configure it for our specifications, since it is primarily aimed at administrators, there is no graphical user interface and all the editing will be done by modifying the `/etc/apt-cacher/apt-cacher.conf`.

So now, open apt-cacher's main configuration file: `/etc/apt-cacher/apt-cacher.conf` and start editing it according to your settings. The default port `apt-cacher` is running on is port `3142`. You might want to change this value accordingly to your needs. You won't typicaly need to bother with this unless, you have a specific service already running on that port.

`allowed_hosts`: by default, all host are allowed to use the repository cache. If you want only a particular IP range to access, you can specify here. If you want to allow over your Wi-Fi say `10.0.0.0/24` and `localhost`, then put the following values

```
allowed_hosts= 10.0.0.0/24
```

notice how localhost is always allowed by default.

Let us forget about the generate_reports option, it is also self explainatory.

Next comes the import setting of `path_map`

`path_map`: This is an interesting directive. Here you can define different aliases for different repository host. So for a typical ubuntu setup, you will have:

```
path_map =  ubuntu archive.ubuntu.com/ubuntu; ubuntu-updates archive.ubuntu.com/ubuntu ; ubuntu-security security.ubuntu.com/ubuntu
```

The mappings above are explained below:

`ubuntu` and `ubuntu-updates` to host `archive.ubuntu.com/ubuntu`

```
and ubuntu-security to security.ubuntu.com
```

Now, in order to access a specific repository, we simply need to append the mapping name to our cache repository server, like: `repository_cache_machine:port/mapping_name`

So, for instance, we can access Ubuntu security repository through **http://repository-cache:3142/ubuntu-security**.

The autostart value of 1, enables the apt-cacher.

```
AUTOSTART=1
```

At this point we should restart the apt-cacher for our changes to take place.

```
$ sudo /etc/init.d/apt-cacher restart.
```

This was the server part of our setup, now login to each of the clients. Don't worry, you don't need to physically move, you can log in via SSH as explained earlier, or you can even write an automated script to do so. Let us update all our clients `/etc/apt/sources.list` files so every host on the network will use our repository-cache machine. It is a good idea to even update the sources.list on the server, as any software downloaded by the server will also be available for rest of the machines.

6.1 Editing the sources.list

Here is a sample original sources.list

```
# deb cdrom:[Ubuntu 10.04 LTS _Lucid Lynx_ - Release amd64 (20100429)]/ lucid main restricted
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.

deb http://lk.archive.ubuntu.com/ubuntu/ lucid main restricted
deb-src http://lk.archive.ubuntu.com/ubuntu/ lucid main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://lk.archive.ubuntu.com/ubuntu/ lucid-updates main restricted
deb-src http://lk.archive.ubuntu.com/ubuntu/ lucid-updates main restricted

## N.B. software from this repository is ENTIRELY
```

```
UNSUPPORTED by the Ubuntu
  ## team. Also, please note that software in universe
WILL NOT receive any
  ## review or updates from the Ubuntu security team.
  deb http://lk.archive.ubuntu.com/ubuntu/ lucid universe
  deb-src  http://lk.archive.ubuntu.com/ubuntu/  lucid
universe
  deb  http://lk.archive.ubuntu.com/ubuntu/  lucid-updates
universe
  deb-src  http://lk.archive.ubuntu.com/ubuntu/  lucid-
updates universe

  ##  N.B.  software  from  this  repository  is  ENTIRELY
UNSUPPORTED by the Ubuntu
  ## team, and may not be under a free licence. Please
satisfy yourself as to
  ## your rights to use the software. Also, please note
that software in
  ## multiverse WILL NOT receive any review or updates
from the Ubuntu
  ## security team.
  deb http://lk.archive.ubuntu.com/ubuntu/ lucid multiverse
  deb-src  http://lk.archive.ubuntu.com/ubuntu/  lucid
multiverse
  deb  http://lk.archive.ubuntu.com/ubuntu/  lucid-updates
multiverse
  deb-src  http://lk.archive.ubuntu.com/ubuntu/  lucid-
updates multiverse

  ## Uncomment the following two lines to add software
from the 'backports'
  ## repository.
  ## N.B. software from this repository may not have been
tested as
  ## extensively as that contained in the main release,
although it includes
  ## newer versions of some applications which may provide
useful features.
  ## Also, please note that software in backports WILL NOT
receive any review
  ## or updates from the Ubuntu security team.
  #   deb   http://lk.archive.ubuntu.com/ubuntu/   lucid-
```

```
backports main restricted universe multiverse
  # deb-src http://lk.archive.ubuntu.com/ubuntu/ lucid-
backports main restricted universe multiverse

  ## Uncomment the following two lines to add software
from Canonical's
  ## 'partner' repository.
  ## This software is not part of Ubuntu, but is offered
by Canonical and the
  ## respective vendors as a service to Ubuntu users.
  # deb http://archive.canonical.com/ubuntu lucid partner
  # deb-src http://archive.canonical.com/ubuntu lucid
partner

  deb http://security.ubuntu.com/ubuntu lucid-security
main restricted
  deb-src http://security.ubuntu.com/ubuntu lucid-security
main restricted
  deb http://security.ubuntu.com/ubuntu lucid-security
universe
  deb-src http://security.ubuntu.com/ubuntu lucid-security
universe
  deb http://security.ubuntu.com/ubuntu lucid-security
multiverse
  deb-src http://security.ubuntu.com/ubuntu lucid-security
multiverse
```

This will become the following, notice how third-party sources won't be touched. In case you are wondering what lk is, it stands for Sri Lanka.

```
  # deb cdrom:[Ubuntu 10.04 LTS _Lucid Lynx_ - Release
amd64 (20100429)]/ lucid main restricted
  # See http://help.ubuntu.com/community/UpgradeNotes for
how to upgrade to
  # newer versions of the distribution.

  deb Insert-Repository-Server-Ip lucid main restricted
  deb-src Insert-Repository-Server-Ip lucid main restricted

  ## Major bug fix updates produced after the final
release of the
  ## distribution.
  deb   Insert-Repository-Server-Ip   lucid-updates   main
restricted
  deb-src Insert-Repository-Server-Ip lucid-updates main
restricted

  ##  N.B.  software  from  this  repository  is  ENTIRELY
UNSUPPORTED by the Ubuntu
  ## team. Also, please note that software in universe
WILL NOT receive any
  ## review or updates from the Ubuntu security team.
  deb Insert-Repository-Server-Ip lucid universe
  deb-src Insert-Repository-Server-Ip lucid universe
  deb Insert-Repository-Server-Ip lucid-updates universe
  deb-src   Insert-Repository-Server-Ip   lucid-updates
universe

  ##  N.B.  software  from  this  repository  is  ENTIRELY
UNSUPPORTED by the Ubuntu
  ## team, and may not be under a free licence. Please
satisfy yourself as to
  ## your rights to use the software. Also, please note
that software in
  ## multiverse WILL NOT receive any review or updates
from the Ubuntu
  ## security team.
  deb Insert-Repository-Server-Ip lucid multiverse
  deb-src Insert-Repository-Server-Ip lucid multiverse
  deb Insert-Repository-Server-Ip lucid-updates multiverse
  deb-src   Insert-Repository-Server-Ip   lucid-updates
multiverse

  ## Uncomment the following two lines to add software
from the 'backports'
  ## repository.
  ## N.B. software from this repository may not have been
tested as
  ## extensively as that contained in the main release,
although it includes
  ## newer versions of some applications which may provide
useful features.
  ## Also, please note that software in backports WILL NOT
receive any review
```

```
  ## or updates from the Ubuntu security team.
  # deb Insert-Repository-Server-Ip lucid-backports main
restricted universe multiverse
  # deb-src Insert-Repository-Server-Ip lucid-backports
main restricted universe multiverse
```

Running the following command after updating the settings will ensure downloading from our local apt-cache.

```
  $sudo apt-get update
```

7 Setting up a web server

A web server is a computer program that delivers (serves) content, such as web pages, using the Hypertext Transfer Protocol (HTTP), over the World Wide Web or a Local Area Network. One of the most popular web servers for Ubuntu is Apache . If you want a quick MySQL and PHP enabled Apache2 server install LAMP.

7.1 Apache

The Apache HTTP Server is one of the most widely used web server. In 2009 it became the first web server software to surpass the 100 million web site milestone. Some of it's features include:

Support for a variety of features, many implemented as compiled modules which extend the core functionality. The modules range from authentication schemes such as mod_access, mod_auth, mod_digest, and mod_auth_digest to server-side programming language support such as PHP, Python and Perl.

SSL support (Using the mod_ssl module)

Proxy module (Using the mod_proxy module)

Serving compressed pages over HTTP.

Configurable error messages.

Virtual hosting: This allows one Apache installation to server many websites. For example one apache installation could simultaneously serve **www.example.com** and **www.example1.com**.

7.2 Installing Apache

You can install Apache from the command line using apt-get or aptitude. It can also be installed using synaptics package manager.

Installing it from the command line is simple:

```
$ sudo aptitude -r install apache
```

This will install Apache2 and recocomended packages to use with the Apache2 web server.

7.3 Configuring Apache

These are the different locations where the files associated with Apache web server are stored in your system:

```
/etc/apache2
```

A directory containing the configuration files for the Apache 2 Web server. The primary configuration file in this directory is the file apache2.conf

```
/etc/apache2/conf.d
```

A directory containing local configuration directives for Apache2, such as those associated with third-party or locally installed packages.

`/etc/apache2/envvars`
A file containing environment variables that you want to set in the environment used by the apache2ctl script to manage an Apache 2 Web server.

`/etc/apache2/mods-available`
A directory containing available Apache 2 modules and their configuration files.

`/etc/apache2/mods-enabled`
A directory containing symbolic links to actively enable Apache 2 modules and their configuration files, located in the /etc/apache2/mods-available directory. This is analogous to the use of symbolic links to start various processes from the scripts in `/etc/init.d` at different run levels.

`/etc/apache2/sites-available`
A directory containing files that define the web sites supported by this server.

`/etc/apache2/mods-enabled`
A directory containing symbolic links to actively enabled

Web sites for this server, located in the `/etc/apache2/mods-available` directory. This is analogous to the use of symbolic links to start various processes from the scripts in `/etc/init.d` at different run levels.

`/etc/default/apache2`
A configuration file that determines whether the Apache 2 should automatically start at boot time.

`/etc/init.d/apache2`
A shell script that uses the apache2ctl utility to start and stop an Apache 2 web server.

`/etc/mime.types`
The default MIME (Multipurpose Internet Mail Extensions) file types and the extensions that they are associated with.

`/usr/lib/cgi-bin`

The location in which any CGI-BIN (Common Gateway Interface scripts) for a default Apache 2 Web server will be installed.

`/usr/sbin/apache2`
The actual executable for the Apache 2 Web server.

`/usr/sbin/apache2ctl`
An administrative shell script that simplifies starting, stopping, restarting, and monitoring the status of a running Apache 2 Web server.

`/usr/share/apache2-doc`
A directory that contains the actual Apache 2 manual (in the manual subdirectory). This directory is present only if you've installed the apache2-doc package (as suggested earlier).

`/usr/share/apache2/error`
A directory containing the default error responses delivered.

`/usr/share/apache2/icons`
A directory containing the default set of icons used by an Apache 2 Web server. This directory is mapped to the `/icons` directory in your Apache server's primary configuration file.

`/var/log/apache2/access.log`
The default access log file for an Apache 2 Web server. This log file tracks any attempts to access this web site, the hosts that they came from, and so on.

`/var/log/apache2/error.log`
The default error log file for an Apache 2 Web server. This log file tracks internal Web server problems, attempts to retrieve nonexistent files, and so on.

`/var/run/apache2/apache2.pid`
A text file used by Apache 2 to record its process ID when it starts. This file is used when terminating or restarting the Apache 2 server using the /etc/init.d/apache2 script.

`/var/www/apache2-default`
A directory containing the default home page for this Web server. Note that the default Apache 2 Web server does not display the content of this directory correctly.
You will be able to access the default Apache webpage at **http://localhost/.**

The main cofiguration file is located at /etc/apache2/apache.conf Default document root for apache2 is /var/www. This means you need to put the files that you want Apache to serve in /var/www. If you want to change the default document root, you need to edit the /etc/apache2/sites-available/default file and look for this line DocumentRoot /var/www/. Here you can change where ever you want to change. For example, if you want to change /home/www the above line looks like this DocumentRoot /home/www/. All the configuration options are well documented and you need to restart the apache daemon whenever you change the config file

```
$ sudo /etc/init/dapache2 restart
```

7.4 LAMP

If you want to get MySql and PHP support for apache, the easist way is to install LAMP:

```
$ sudo apt-get install lamp-server^
```

The ^ is not a typo.

After the packages get downloaded and installed, you will be asked to change the root user's password on the MySql database. After this, the rest of the packages get installed. Now you need to test if all the components are working.

7.5 Testing Apache

Acess **http://localhost/** from a browser. You should see a web page that says `It Works!`

7.6 Testing PHP

First create a simple php file in /var/www :

```
$ sudo echo “<?php phpinfo(); ?>” > /var/www/test.php
```

and restart apache

```
$ sudo /etc/init.d/apache2 restart
```

Now navigate to **http://localhost/test.php/.** You should see a page describing version information for your php installation.

7.7 Installing PhpMyAdmin

phpMyAdmin is not essential, but it is a much simpler way to get in and adjust things in your MySQL database if you are not familiar with MySQL's commands. You can install phpMyAdmin using apt-get:

```
$ sudo apt-get install libapache2-mod-auth-mysql phpmyadmin
```

The installation will ask you to select a webserver for automatic configuration. Make sure you choose apache2. Follow the installer and it will install and configure phpMyAdmin for you.

You can access phpMyAdmin by pointing your browser to **http://localhost/phpmyadmin** and logging in using your MySQL root password.

You can now put your web site's files into `/var/www/site1` (change site1 to your web site's name).

You can access your web site by pointing your browser to **http://localhost/site1.**

7.8 Setting up a mail server

To set up a mail server you need to install various packages:
Mail Transfer Agent: Postfix
A MTA transfers email from one computer to another.
POP/IMAP: Courier IMAP
An application-layer internet standart protocool used by local e-mail clients to retreive e-mail from a remote server over TCP/IP
Database: MySQL
MySQL is well supported for the sort of lookups required in a mail server.
Anti-Spam: SpamAssassin
Spam Assasin is a Powerfull renowned spam fighting tool.
Anti-Virus: ClamAV
Clam AV is a free virus scanner that includes an update daemon
Spam Greylisting: PostGrey
Postgrey is an excellent little script to stop 99 per cent of all spam. All it does is on first contact for specific from-to combinations, tells the sender server to try again in a little while, which most spammers cant afford to do. When proper servers try again after a few minutes it lets it through.
WebMail: SquirrelMail or Roundcube
Squirrel mail is an easy to set up php based web mail client with an extensive plugin selection.

7.9 Installing the packages

First make sure that your package sources are currently pointing to main,multiverse,restricted and universe and then update your system.

Then, install MySQL if you don't already have it:

```
$ sudo aptitude install mysql-client mysql-server
```

Then, install postfix:

```
$ sudo aptitude install postfix postfix-mysql
```

This will prompt you to choose the type of email server, choose it as per your requirements.

Install ClamAV:

```
$ sudo aptitude install clamav-base libclamav6 clamav-daemon clamav-freshclam
```

Install SpamAssassin:

```
$ sudo aptitude install spamassassin spamc
```

Install PostGrey:

```
$ sudo aptitude install postgrey
```

Install SquirrelMail:

```
$ sudo aptitude install squirrelmail squirrelmail-locales php-pear php5-cli
```

Install phpMyAdmin:

```
$ sudo aptitude install phpmyadmin
```

7.10 Configure various components

Postfix

You should put the name of your server in this file /etc/mailname, It can be something like smtp.domain.name, where domain name is your domain name.

Now edit the main postfix configuration file:

```
$ sudo gedit /etc/postfix/main.cf
```

Ubuntu already puts in some sensible default values in this file. You may need to comment some of them.

First specify the name of your server.

```
 # This is already done in /etc/mailname
 #myhostname= mail.domain.com
```

Next is the origin which is the domain appended to email from this machine, this can be your full servername, or domain name.

```
# myorigin=/etc/mailname
myorigin=domain.com
```

Then decide what the greeting text will be. Enough info so it is useful, but not divelge everything to potential hackers.

```
smtpd_banner = $myhostname ESMTP $mail_name
```

Next you need to decide whether to send all outgoing mail via another SMTP server, or send them yourself. If send via my an external server it has to worry about the queing etc. If you send it yourself then you are not reliant on 3rd party server. But you may risk more exposure and accidentally be blocked by spam blockers. And it is more work for your server. Also many servers block dynamic dns hosts, so you may find your server gets rejected. However choose whichever you are suits your needs better

```
 # leave blank to do it yourself relayhost =
# or put it an accessible smtp server
relayhost = smtp.externalserver.com
```

Next is network details. You will accept connection from anywhere, and you only trust this machine inet_interfaces = all
mynetworks_style = host

Next you can masquerade some outgoing addresses. Say your machine's name is mail.domain.com. You may not want outgoing mail to come from username@mail.example.com, as you'd prefer username@example.com. You can also state which domain not to masquerade. E.g. if you use a dynamic dns service, then your server address will be a subdomain. You can also specify which users not to masquerade.

```
 # masquerade_domains = mail.example.com !www.example.
comsub.dyndomain.com
# masquerade_exceptions = root
```

As we will be using virtual domains, these need to be empty.

```
local_recipient_maps =
mydestination =
```

Then will set a few numbers.

```
# how long if undelivered before sending warning update
to sender
 delay_warning_time = 4h
 # will it be a permanent error or temporary
 unknown_local_recipient_reject_code = 450
# how long to keep message on queue before return as
failed.
maximal_queue_lifetime = 7d
# max and min time in seconds between retries if
connection failed
minimal_backoff_time = 1000s maximal_backoff_time =
8000s
# how long to wait when servers connect before receiving
rest of data
smtp_helo_timeout = 60s
# how many address can be used in one message.
```

```
  # effective stopper to mass spammers, accidental copy
in whole address list
  # but may restrict intentional mail shots.
  smtpd_recipient_limit = 16
  # how many error before back off.
  smtpd_soft_error_limit = 3
  # how many max errors before blocking it.
  smtpd_hard_error_limit = 12
```

Now we can specify some restrictions. Be carefull that each setting is on one line only.

```
   # Requirements for the HELO statement
   smtpd_helo_restrictions = permit_mynetworks, warn_if_
reject reject_non_fqdn_hostname, reject_invalid_hostname,
permit
  # Requirements for the sender details
  smtpd_sender_restrictions = permit_mynetworks, warn_
if_reject reject_non_fqdn_sender, reject_unknown_sender_
domain, reject_unauth_pipelining, permit
  # Requirements for the connecting server smtpd_client_
restrictions = reject_rbl_client sbl.spamhaus.org, reject_
rbl_client blackholes.easynet.nl, reject_rbl_client dnsbl.
njabl.org
  # Requirement for the recipient address
  smtpd_recipient_restrictions = reject_unauth_pipelining,
permit_mynetworks, reject_non_fqdn_recipient, reject_
unknown_recipient_domain, reject_unauth_destination,
permit smtpd_data_restrictions = reject_unauth_pipelining
```

Further restrictions:

```
   # require proper helo at connections
  smtpd_helo_required = yes
  # waste spammers time before rejecting them
  smtpd_delay_reject = yes
  disable_vrfy_command = yes
```

Next we need to set some maps and lookups for the virtual domains.

```
  alias_maps = hash:/etc/postfix/aliases
  alias_database = hash:/etc/postfix/aliases
  # this specifies where the virtual mailbox folders will
be located
  virtual_mailbox_base = /var/spool/mail/virtual # this is
```

```
for the mailbox location for each user virtual_mailbox_
maps = mysql:/etc/postfix/mysql_mailbox.cf # and this
is for aliases virtual_alias_maps = mysql:/etc/postfix/
mysql_alias.cf # and this is for domain lookups virtual_
mailbox_domains = mysql:/etc/postfix/mysql_domains.cf
  # this is how to connect to the domains (all virtual,
but the option is there)
  # not used yet #
  transport_maps = mysql:/etc/postfix/mysql_transport.cf
```

You can use a lookup for the uid and gid of the owner of mail files.

```
  virtual_uid_maps = static:5000 virtual_gid_maps =
static:5000
```

You need to set up an alias file. This is only used locally, and not by your own mail domains.

```
  $  sudo cp /etc/aliases /etc/postfix/aliases
  # may want to view the file to check if ok.
  # especially that the final alias, eg root goes
  # to a real person
  $sudo postalias /etc/postfix/aliases
```

Next you need to set up the folder where the virtual mail will be stored. This may have already been done by the apt-get. And also create the user who will own the folders.

```
   # to add if there is not a virtual user
  sudo mkdir /var/spool/mail/virtual
  sudo groupadd --system virtual -g 5000
  sudo useradd --system virtual -u 5000 -g 5000
  sudo chown -R virtual:virtual /var/spool/mail/virtual
```

7.11 Postfix's MySQL configuration

Next we need to set up the files to access the lookups via the database. We will only set up a few now, and the rest later when/if needed:

Edit or create `mysql_mailbox.cf` which tells postfix how to find the users mailbox location

```
  $ sudo gedit /etc/postfix/mysql_mailbox.cf
  user=mail
  password=mailPASSWORD
  dbname=maildb
  table=users
  select_field=maildir
  where_field=id hosts=127.0.0.1
  additional_conditions = and enabled = 1
```

Create the file which tells postfix how to find the email alias:

```
  $ sudo gedit /etc/postfix/mysql_alias.cf
  user=mail password=mailPASSWORD
  dbname=maildb
  table=aliases select_field=destination
  where_field=mail  hosts=127.0.0.1  additional_conditions
= and enabled = 1
```

Create the file which tells postfix how to find the domains:

```
  $ sudo gedit /etc/postfix/mysql_domains.cf
  user=mail password=mailPASSWORD
   dbname=maildb
   table=domains
   select_field=domain where_field=domain
  hosts=127.0.0.1 additional_conditions = and enabled = 1
```

If you specify an IP in hosts, (as opposed to 'localhost') then it will communicate over tcp and not the mysql socket due to a `chroot` restriction. Don't forget to replace the passwords with your chosen mail user password.

7.12 Database

MySQL

Now we will need to create the tables for thos lookups just specified. First you need to create a user to use in MySQL for mail only. Then you need to create the database, Take note of your chosen mail username and password. You will need the password you specified for root during MySQL package installation.

```
  # If not already done (in package installation or
previous configuration)...
  mysqladmin -u root password new_password
  # log in as root mysql -u root -p
  # then enter password for the root account when prompted
  Enter password:
  # then we create the mail database create database
maildb;
  # then we create a new user: "mail"
  GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON maildb.*
TO 'mail'@'localhost' IDENTIFIED by 'mailPASSWORD';
  GRANT   SELECT,INSERT,UPDATE,DELETE,CREATE,DROP   ON
```

```
maildb.* TO 'mail'@'%' IDENTIFIED by 'mailPASSWORD'; exit;
```

Replace mailPASSWORD with your chosen password.

Then you will need to create these tables:

```
aliases
domains
users
```

You will need to create more later on for the extensions, but only these are relevant now.

Log in to mysql as the new mail user

```
 mysql -u mail -p maildb
# enter the newly created password
 Enter password:
```

Then run this commands to create the tables:

```
CREATE TABLE `aliases` (
 `pkid` smallint(3) NOT NULL auto_increment,
 `mail` varchar(120) NOT NULL default ',
 `destination` varchar(120) NOT NULL default '',
 `enabled` tinyint(1) NOT NULL default '1',
 PRIMARY KEY (`pkid`),
 UNIQUE KEY `mail` (`mail`)
 ) ;

CREATE TABLE `domains` (
 `pkid` smallint(6) NOT NULL auto_increment,
 `domain` varchar(120) NOT NULL default '',
 `transport` varchar(120) NOT NULL default 'virtual:',
 `enabled` tinyint(1) NOT NULL default '1',
 PRIMARY KEY (`pkid`)
 ) ;

CREATE TABLE `users` (
 `id` varchar(128) NOT NULL default '',
 `name` varchar(128) NOT NULL default '',
 `uid` smallint(5) unsigned NOT NULL default '5000',
 `gid` smallint(5) unsigned NOT NULL default '5000',
 `home` varchar(255) NOT NULL default '/var/spool/mail/
virtual',
  `maildir` varchar(255) NOT NULL default 'blah/',
`enabled` tinyint(3) unsigned NOT NULL default '1',
`change_password` tinyint(3) unsigned NOT NULL default
```

'1', `clear` varchar(128) NOT NULL default 'ChangeMe', `crypt` varchar(128) NOT NULL default 'sdtrusfX0Jj66', `quota` varchar(255) NOT NULL default '', `procmailrc` varchar(128) NOT NULL default '', `spamassassinrc` varchar(128) NOT NULL default '', PRIMARY KEY (`id`), UNIQUE KEY `id` (`id`)) ;

The last few fields in the users table are not required, but useful if you extend later.

# To visualise the tables created: describe aliases; describe domains; describe users; # then quit mysql exit;

Next, edit the MySQL's `my.cnf` file. In Ubuntu/Debian this is created by default. In Mandrake we had to manually create a blank one in /etc. But we need to configure it, so: `sudo vi /etc/mysql/my.cnf`. In previous version you needed to comment out this line #skip-networking However in todays file the default is to bind the address to localhost, which is fine. `bind-address = 127.0.0.1`. It is very useful at the start to log any SQL calls that makes it to MySQL. So enable these lines: `general_log_file = /var/log/mysql/mysql.log general_log = 1`. Then in a few weeks comment it out when everything is working, as it slows mysql down

Restart MySQL to make sure its picking up the new settings. `sudo /etc/init.d/mysql restart`

Return to top.

Pop/IMAP

Courier IMAP

```
$ sudo gedit /etc/courier/authdaemonrc
```

Change to mysql mode.

```
authmodulelist="authmysql"
```

Further down enable logging.

```
DEBUG_LOGIN=2
sudo gedit /etc/courier/authmysqlrc
MYSQL_USERNAME mail
```

Change the password to whichever you have chosen

```
MYSQL_PASSWORD mailPASSWORD
MYSQL_DATABASE maildb
MYSQL_USER_TABLE users
MYSQL_CRYPT_PWFIELD crypt
# MYSQL_CLEAR_PWFIELD clear
MYSQL_MAILDIR_FIELD concat(home,'/',maildir)
 MYSQL_WHERE_CLAUSE enabled=1
```

Lastly you can have a look at the imapd file, but no changes is needed. It is located in `/etc/courier/imapd`

7.13 Antispam

The default config of spam assassin is ok. But you need to tell SpamAssassin to start smapd at boot.

```
$ sudo gedit /etc/default/spamassassin.
```

You can also enable Bayes and auto learning by editing `/etc/spamassassin/local.cf`

7.14 Antivirus

ClamAV has sensible defaults except the fact that it updates 24 times a day, If you want to change that edit the configuration file in `/etc/clamav`.

7.15 PostGrey

Postgrey has a good default configuration but you need to tell Postfix to use it.

Edit postfix's main config:

```
$ sudo gedit /etc/postfix/main.cf
```

And then edit the recipient restrictions:

```
smtpd_recipient_restrictions = reject_unauth_pipelining, permit_mynetworks, permit_sasl_authenticated, reject_non_fqdn_recipient, reject_unknown_recipient_domain, reject_unauth_destination, check_policy_service inet:127.0.0.1:10023, permit
```

You can tweak options such as delay and auto whitelisting and reject message in `/etc/default/postgrey`.

7.16 Webmail

This tutorial shows you how to set up squirrelmail instead of roundcube because roundcube is still in beta.

To run Squirrelmails configuration utility run:

```
$ sudo squirrelmail-configure
```

You need to configure apache now, run this command:

```
$ sudo a2ensite squirrelmail
```

then reload apache configuration:

```
$ sudo /etc/init.d/apache2 force-reload
```

Now your mail server is almost setup, you still need to add users and domains.

Before adding your own users and domains, you need to add some required default data.

Use phpMyAdmin or command line mysql to do this:

```
INSERT INTO domains (domain) VALUES
                   ('localhost'),
                   ('localhost.localdomain');
```

These are some default mail aliases:

```
INSERT INTO aliases (mail,destination) VALUES
              ('postmaster@localhost','root@localhost'),
     ('sysadmin@localhost','root@localhost'),
     ('webmaster@localhost','root@localhost'),
     ('abuse@localhost','root@localhost'),
     ('root@localhost','root@localhost'),
     ('@localhost','root@localhost'),
     ('@localhost.localdomain','@localhost');
```

Then insert a root user:

```
  INSERT INTO users (id,name,maildir,crypt) VALUES
('root@localhost','root','root/', encrypt('apassword') );
```

You can now now add more users or domains based on your needs.

8 Memory Management

Linux was designed in the days when RAM was expensive, and it still is as compared to hard disk drive. To use a portion of your hard disk drive as a piece of RAM, Linux supports Virtual Memory. Linux kernel will keep looking for currenly unused block of RAM and will transport them to the Virtual Memory to keep RAM free and available for the task at hand. To get speed advantages even with the hard disk drive, it is always recommended to maintain a separate partition for swap.

The swap file is just like any other file on your system as far asthe kernel is concerned wth the only restriction that it must be available locally and not mounted over the NFS.

You should also know that Linux allows one to use several swap partitions and/or swap files at the same time. This means that if you only occasionally need an unusual amount of swap space, you can set up an extra swap file at such times, instead of keeping the whole amount allocated all the time.

Just to be clear swapping and paging are almost used interchangibiliy as terms but the difference is that that swapping involves writing the whole process out, while paging indicates towards writing only specific blocks in and out.

Creating a swap file is a simple two step process, one is creating space for it and the second being to indicate it as swap.

The first part can be done as follows:

```
$ dd if=/dev/zero of=/digit-swap bs=1024
        count=1024
        1024+0 records in
        1024+0 records out
```

1024 above is the size in kilobytes. Digit-swap is of course the name of the swap file, the point to note here is that the size should always be a multiple of 4, as the kernel writes out memory pages which are 4 kilobytes in size, so this ensures maximum efficency. Now to mark it as swap, we use the command `mkswap`, the procedure is illustrated below:

```
$ mkswap /digit-swap 1024
        Setting up swapspace, size = 1044480
        bytes
```

At this point it is critical to note that we have created the swap space and is ready to be used but the kernel will still not use it, as it is still not instructed to do so, this is analogous to having a pen drive with you but not inserting it into the USB drive. To use the file as swap, you should instruct the kernel explicitly using the command `swapon`, an example below

```
$ swapon /digit-swap
```

This will make the file `digit-swap` in the root directory as an active swap space. The swap files are hot pluggable, which means you don't need to restart the system.

Now doing a swapon everytime your computer starts is a very cumbersome method, so we will create an entry into the `/etc/fstab` file so that our swap is mounted automatically on boot.

So the entries as follow in your `/etc/fstab`

```
/dev/sda4       none      swap      sw    0    0
/swapfile       none      swap      sw    0    0
```

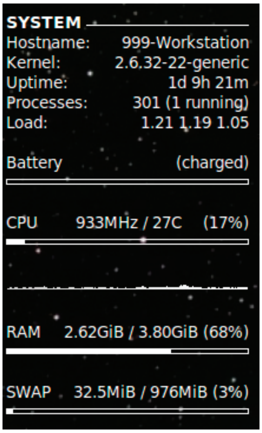
indicate a whole partition `/dev/sda4` and a file `digit-swap` being used as swap. A quick tip is that partition names starting wth sda are typically SATA disk partitions and hda are your older PATA disk partitions.

To monitor your current ram usage and swap usage, you can always issue the command as follows:

```
$ free
```

this will produce the output of following format:

```
                total        used         free       shared
buffers     cached
  Mem:        3989104     3564764      424340            0
229092      592048
  -/+ buffers/cache:     2743624     1245480
  Swap:        999416       33276      966140
```



```
SYSTEM
Hostname:      999-Workstation
Kernel:       2.6.32-22-generic
Uptime:              1d 9h 21m
Processes:      301 (1 running)
Load:            1.21 1.19 1.05

Battery               (charged)

CPU     933MHz / 27C     (17%)

RAM   2.62GiB / 3.80GiB (68%)

SWAP   32.5MiB / 976MiB (3%)
```

Remember that since swap is used only when the operating system is running, if you have multiple installations of various variants of linux on your system, you can use the same swap partition for all of them, rather it will be a recommended practice.

It's a good idea to have at least some swap space, even if you think that you have enough ram to not need one. Linux uses swap space somewhat aggressively, so that as much physical memory as possible can be kept free. Linux will swap out memory pages that have not been used, even if the

A graphical way, using uber-cool conky to monitor your usage memory is not yet needed for anything. This avoids waiting for swapping when it is needed: the swapping can be done earlier, when the disk is otherwise idle.

9 Automation

As the number of systems you manage increase, the amount of time you would spent (read waste) doing the same thing would increase exponentially, making the process of automating you Linux Administration task a necessity. Along side the major benefit of saving time, automation brings along advantages like consistency in your setup, regularity in mundane but critical tasks such as backup, compliance with standards and guidelines and the last but not the least, you get a homogeneous and easy to maintain setup. Likewise, you might think you don't need automation if you have only one server in your company or home. However, you might want it because backups and timely security updates are easy tasks for a busy system administrator to neglect, even in this most basic setup. Automation is already a core part of UNIX philosophy, and cron jobs have historically been the de facto method for automating UNIX tasks.

An important part of the automation is the ability to access all your systems from a single location and execute your commands safely, reliably and securely on it. For that reason, we will now introduce SSH, it is like a tool, what you do with it is completely up to you, reading the Appendix on "BASH Shell scripting" will prove immensely helpful.

Install the necessary tool by issuing the following command:

```
$ sudo apt-get install openssh-client openssh-server
```

To begin with, try logging into your own machine by issuing the following command:

```
$ ssh your_username@your_ip
```

Following this, you will be asked for your password and issued several warnings. Ignore them for now, we will come back to them. Don't fret.

You are now logged on to your own terminal via SSH, and you can do anything you want as if it were a native terminal. You can, of course, login to any remote machine via the same syntax.

```
$ ssh remote_machine_user_name@remote_machine_IP
```

So far so good, but till now your local and remote environment were isolated entities. To transfer the message between the matrix, use SCP, or secure copy.

```
$ scp local_source user@host:/remote_dest
$  scp  userA@hostA:/remote_fileA  userB@hostB:/remote_
```

```
fileB
```

The last example, there is no local instance, you are directly operating between two remote locations.

9.1 Public-Key Authentication

You must have wondered at this point that during the automation procedure how exactly do we plan to enter the passwords without human interventation, of course we can store them as default values in our text script, but then we are storing our sensitive passwords in text files that can be accessed by anyone. To counter this, we use Public-Key Authentication systems.

We begin by generating the key pair

```
$ shh-keygen -t rsa -b 2048 -c 'digit@testkey.com'
```

It will ask you for a passphrase twice. Fancy yourself, as for the location, the default will be .ssh in your home directory.

The two files containing your fingerprint will be `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`

As you can see, adjusting the `-b` parameter would adjust the size, you can put `8192` if you are a little concerned. The `-c` option is for comment, so you can distinguish the needed key pair from others.

The file with the extension `*.pub` is your public key, placing it on the machine which is running the ssh server, would then allow you easy access courtesy your private key. Do the following to ensure correct permissions on the server

```
$ mkdir -p ~/.ssh
```

assuming it ain't thereafter

```
$chmod 0700 ~/.ssh
```

bullet proofing the permissions.

```
$ cp your_pub_key_file_location ~/.ssh/authorized_keys

$chmod 0600 ~/.ssh/authorized_keys
```

for adding more keys, you can keep appending to the file.

So basically all your clients will generate the key pair, and then you can add their public key to the server and vice-versa so both can access each other in a secured and authorised manner.

9.2 SSH-agent

Currently in the above scheme there is a small problem, your private key is still left unencrypted, so we have made our operations passwordless, but the private key is still unencrypted, to counter this we will use ssh-agent, which is a progrem that lets you enter your passphrase only once per session and then decrypts your private key, but stores it only in the memory.

Issue the following in the terminal

```
$ ssh-agent bash
```

Of course if you want a more graphical inteface you can do that with the awesomeness of `screen`. If it isn't installed, you can install it as follows:

```
$ sudo apt-get install screen
```

and then you have the option of

```
$ ssh-agent screen.
```

Now you should add your private key(s) with ssh-add:

```
$ ssh-add
```

It will find your ssh keys, and ask for passphrase for each, after a successful operation it will announce via a `Identity added:` message.

So your ssh-agent and your session will last till you keep your terminal window/session open or explicitly close the ssh-agent process. Of course there are many more tricks with ssh, but they are beyond the scope of this introductory guide. So the above introduction to SSH, combined with the skill of shell programming in the Appendix, leaves you with the tools to manipulate in batch even your remote machines.

10 Backups

We will now discuss the process of creating snapshot style backups. Snapshot backups are a feature of some high-end industrial file servers; they create the illusion of multiple, full backups per day without the space or processing overhead. It is often possible to store several weeks worth of snapshots with slightly more than twice the original storage. We will stick to the basics and only use file utilities and another tool called `rsync`. The advantage is obvious enough, you can revert to a snapshot in case of data loss or a miscalculated configuration change.

10.1 Using rsync to make a backup

Rsync is your swiss army knife for backups, not only is it full of features, it is also very polished and powerful. Let us learn it in detail, you will find its applications in areas even other than backup, including monitoring resources.

Covering the obvious parts first, assume you have a directory called source, and you want to back it up into the directory destination.

In the most simple manner, you will just need to issue the following

```
$ rsync -a source/ destination/
```

you can add `-v` or `-vv` to adjust the verbosity of the output according to you, at this point you might be wondering how is this any different from the following:

```
$ cp -a source/. Destination/
```

Now as a matter of fact there won't be any difference in the first run, but following that only changes get updated, so you can sync say 100 GB of records in which only a few MB change everyday in 2 minutes or less resulting in high efficiency.

We have convered ssh in great detail above, what is the point if we don't apply it here, so:

```
rsync -a -e ssh source/ username@remotemachine.com:/
path/to/destination/
```

10.2 Confusion around trailing slashes

You may be accustomed to commands that don't care about trailing slashes. For example, if a and b are two directories, then cp -a a b is equivalent to cp -a a/ b/. However, rsync does care about the trailing slash, but only on the source argument. For example, let a and b be two directories, with the file foo initially inside directory a. Then type this command:

```
rsync -a a b
```

produces `b/a/foo`, whereas the command:

```
rsync -a a/ b
```

produces `b/foo`. The presence or absence of a trailing slash on the destination argument (b, in this case) has no effect.

10.3 Using the --delete flag

If a file was originally in both source/ and destination/ (from an earlier rsync, for example), and you delete it from source/, you probably want it to be deleted from destination/ on the next rsync. However, the default behavior is to leave the copy at destination/ in place. Assuming you want rsync to delete any file from destination/ that is not in source/, you'll need to use the `--delete` flag:

```
rsync -a --delete source/ destination/
```

10.4 Be lazy: use cron

One of the toughest obstacles to a good backup strategy is human nature; if there's any work involved, there's a good chance backups won't happen. Fortunately, there's a way to harness human laziness: make cron do the work.

To run the rsync-with-backup command from the previous section every morning at 4:20 AM, for example, edit the root cron table: (as root)

```
crontab -e
```

Then add the following line

```
20 4 * * * rsync -a --delete source/ destination/
```

Finally, save the file and exit. The backup will happen every morning at precisely 4:20 AM, and root will receive the output by email. Don't copy that example verbatim, though; you should use full path names (such as `/usr/bin/rsync` and `/home/source/`) to remove any ambiguity.

10.5 Incremental backups with rsync

Since making a full copy of a large filesystem can be a time-consuming and expensive process, it is common to make full backups only once a week or once a month, and store only changes on the other days. These are called "incremental" backups, and are supported by the venerable old dump and tar utilities, along with many others.

However, you don't have to use tape as your backup medium; it is both possible and vastly more efficient to perform incremental backups with `rsync`.

The most common way to do this is by using the `rsync -b --backup-dir=` combination. We've seen examples of that usage here, but we won't discuss it further, because there is a better way. If you're not familiar with hard links, though, you should first start with the following review.

10.6 Review of hard links

We usually think of a file's name as being the file itself, but really the name is a hard link. A given file can have more than one hard link to itself – for example, a directory has at least two hard links: the directory name and . (for when you're inside it). It also has one hard link from each of its sub-directories (the .. file inside each one). If you have the stat utility installed on your machine, you can find out how many hard links a file has (along with a bunch of other information) with the command:

```
stat filename
```

Hard links aren't just for directories – you can create more than one link to a regular file too. For example, if you have the file a, you can make a link called b:

```
ln a b
```

Now, a and b are two names for the same file, as you can verify by seeing that they reside at the same inode (the inode number will be different on your machine):

```
ls -i a
  232177 a
ls -i b
  232177 b
```

So `ln a b` is roughly equivalent to `cp a b`, but there are several important differences:

1. The contents of the file are only stored once, so you don't use twice the space.
2. If you change a, you're changing b, and vice-versa.
3. If you change the permissions or ownership of a, you're changing those of b as well, and vice-versa.
4. If you overwrite a by copying a third file over it, you will also overwrite b, unless you tell cp to unlink before overwriting. You do this by running cp with the `--remove-destination` flag. Notice that `rsync` always unlinks before overwriting. Note, added 2002.Apr.10: the previous statement applies to changes in the file contents only, not permissions or ownership.

But this raises an interesting question. What happens if you rm one of the links? The answer is that rm is a bit of a misnomer; it doesn't really remove a file, it just removes that one link to it. A file's contents aren't truly removed until the number of links to it reaches zero. In a moment, we're going to make use of that fact, but first, here's a word about cp.

10.7 Using cp -al

In the previous section, it was mentioned that hard-linking a file is similar to copying it. It should come as no surprise, then, that the standard GNU `coreutils cp` command comes with a `-l` flag that causes it to create (hard) links instead of copies (it doesn't hard-link directories, though, which is good; you might want to think about why that is). Another handy switch for the `cp` command is `-a` (archive), which causes it to recurse through directories and preserve file owners, timestamps, and access permissions.

Together, the combination `cp -al` makes what appears to be a full copy of a directory tree, but is really just an illusion that takes almost no space. If we restrict operations on the copy to adding or removing (unlinking) files – i.e., never changing one in place--then the illusion of a full copy is complete. To the end-user, the only differences are that the illusion-copy takes almost no disk space and almost no time to generate.

2002.05.15: Portability tip: If you don't have GNU cp installed (if you're using a different flavour of *nix, for example), you can use find and cpio instead. Simply `replace cp -al a b` with `cd a && find . -print | cpio -dpl ../b`. Thanks to Brage Førland for that tip.

10.8 Putting it all together

We can combine `rsync` and `cp -al` to create what appear to be multiple full backups of a file system without taking multiple disks' worth of space. Here's how, in a nutshell:

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
cp -al backup.0 backup.1
rsync -a --delete source_directory/  backup.0/
```

If the above commands are run once every day, then `backup.0`, `backup.1`, `backup.2`, and `backup.3` will appear to each be a full backup of `source_directory/` as it appeared today, yesterday, two days ago, and three days ago, respectively – complete, except that permissions and ownerships in old snapshots will get their most recent values (thanks to J.W. Schultz for pointing this out). In reality, the extra storage will be equal to the current size of `source_directory/` plus the total size of the changes over the last three days – exactly the same space that a full plus daily incremental backup with dump or tar would have taken.

Update (2003.04.23): As of rsync-2.5.6, the – link-dest flag is now standard. Instead of the separate cp -al and rsync lines above, you may now write:

```
mv backup.0 backup.1
rsync -a --delete --link-dest=../backup.1 source_
directory/  backup.0/
```

This method is preferred, since it preserves original permissions and ownerships in the backup. However, be sure to test it – as of this writing some users are still having trouble getting --link-dest to work properly. Make sure you use version 2.5.7 or later.

Update (2003.05.02): John Pelan writes in to suggest recycling the oldest snapshot instead of recursively removing and then re-creating it. This should make the process go faster, especially if your file tree is very large:

```
mv backup.3 backup.tmp
mv backup.2 backup.3
mv backup.1 backup.2
mv backup.0 backup.1
mv backup.tmp backup.0
cp -al backup.1/. backup.0
rsync -a --delete source_directory/ backup.0/
```

2003.06.02: OOPS! Rsync's link-dest option does not play well with J. Pelan's suggestion--the approach we previously had written above will result in unnecessarily large storage, because old files in `backup.0` will get replaced and not linked. Please only use Dr. Pelan's directory recycling if you use the separate `cp  -al` step; if you plan to use `--link-dest`, start with `backup.0 empty and pristine`.

10.9 Isolating the backup from the rest of the system

If you take the simple route and keep your backups in another directory on the same file system, then there's a very good chance that whatever damaged your data will also damage your backups. In this section, we identify a few simple ways to decrease your risk by keeping the backup data separate.

10.10 The easy (bad) way

In the previous section, we treated /destination/ as if it were just another directory on the same filesystem. Let's call that the easy (bad) approach. It works, but it has several serious limitations:

- If your filesystem becomes corrupted, your backups will be corrupted too.
- If you suffer a hardware failure, such as a hard disk crash, it might be very difficult to reconstruct the backups.
- Since backups preserve permissions, your users--and any programs or viruses that they run--will be able to delete files from the backup. That is bad. Backups should be read-only.
- If you run out of free space, the backup process (which runs as root) might crash the system and make it difficult to recover.
- The easy (bad) approach offers no protection if the root account is compromised.

Keep it on a separate partition

If your backup directory is on a separate partition, then any corruption in the main filesystem will not normally affect the backup. If the backup process runs out of disk space, it will fail, but it won't take the rest of the system down too. More importantly, keeping your backups on a separate partition means you can keep them mounted read-only; we'll discuss that in more detail in the next chapter.

Keep that partition on a separate disk

If your backup partition is on a separate hard disk, then you're also protected from hardware failure. That's very important, since hard disks always fail eventually, and often take your data with them. An entire industry has formed to service the needs of those whose broken hard disks contained important data that was not properly backed up.

Important: Notice, however, that in the event of hardware failure you'll still lose any changes made since the last backup. For home or small office users, where backups are made daily or even hourly as described in this document, that's probably fine, but in situations where any data loss at all would be a serious problem (such as where financial transactions are concerned), a RAID system might be more appropriate.

RAID is well-supported under Linux, and the methods described in this document can also be used to create rotating snapshots of a RAID system.

Keep that disk on a separate machine

If you have a spare machine, even a very low-end one, you can turn it into a dedicated backup server. Make it standalone, and keep it in a physically separate place – another room or even another building. Disable every single remote service on the backup server, and connect it only to a dedicated network interface on the source machine.

On the source machine, export the directories that you want to back up via read-only NFS to the dedicated interface. The backup server can mount the exported network directories and run the snapshot routines discussed in this article as if they were local. If you opt for this approach, you'll only be remotely vulnerable if:

1. a remote root hole is discovered in read-only NFS, and
2. the source machine has already been compromised.

We'd consider this "pretty good" protection, but if you're (wisely) paranoid, or your job is on the line, build two backup servers. Then you can make sure that at least one of them is always offline.

If you're using a remote backup server and can't get a dedicated line to it (especially if the information has to cross somewhere insecure, like the public internet), you should probably skip the NFS approach and use `rsync`

`-e ssh` instead.

It has been pointed out that rsync operates far more efficiently in server mode than it does over NFS, so if the connection between your source and backup server becomes a bottleneck, you should consider configuring the backup machine as an rsync server instead of using NFS. On the downside, this approach is slightly less transparent to users than NFS – snapshots would not appear to be mounted as a system directory, unless NFS is used in that direction, which is certainly another option (We haven't tried it yet though). Thanks to Martin Pool, a lead developer of rsync, for making me aware of this issue.

Here's another example of the utility of this approach – one that we use. If you have a bunch of Windows desktops in a lab or office, an easy way to keep them all backed up is to share the relevant files, read-only, and mount them all from a dedicated backup server using SAMBA. The backup job can treat the SAMBA-mounted shares just like regular local directories.

Run it all with cron

Cron is an administrator's dream, it is simple, fast and accurate task scheduler.

To schedule any task to run once at a later time or recurrently as in the case of our backup, we can schedule it via cron. The first step towards that would be to edit the crontab file. The crontab file is the configuration file for cron jobs, we will edit the crontab of root by opening the file

```
$ vi /etc/crontab
```

The above is the system wide cron file and you would need root permissions to edit it.

To list the cronjobs for current user, issue the following command

```
$ crontab -l
```

A little introduction to crontab format should be of help here.

A cron job consists out of six fields:

```
<minute> <hour> <day of month> <month> <day of week>
<command>
```

```
        field           allowed values
        -----           --------------
        minute          0-59
        hour            0-23
        day of month    1-31
        month           1-12 (or names, see below)
        day of week     0-7 (0 or 7 is Sun, or
use names)
```

When specifying day of week, both day 0 and day 7 will be considered Sunday. A field may be an asterisk (*), which will imply a wildcard value. A / indication execution should happen recurrently.

To make the automatic snapshots happen, you should add the following lines to root's crontab file:

```
0 18 * * *  /usr/local/bin/daily_snapshot_rotate.sh
```

They cause `make_snapshot.sh` to be run every four hours on the hour and `daily_snapshot_rotate.sh` to be run every day at 18:00 (that is, 6:00 PM, say when your office closes).

11 Media Servers

This section is specifically very useful for home users who have multiple machines running typically on Wi-Fi backbone.

Let us first answer an increasingly important question, why would you need a media server and how would it differ from a file server if centralisation is the only concern. Let us begin by answering the first question, a centralised location for all your media prevents you from maintain a copy on each machine and syncing them recurrently with addition of new media. Coming to the second question, it is far different and sophisticated than a simple file server as first and foremost it supports streaming (think YouTube), so you don't have to first download a file, watch/listen to it and then delete it, you can simply stream it, and if you implement QoS in your nework router you are assured of your bandwidth also. Most media severs support UPnP protocol which enables other network enabled media devices to connect and interact. Universal Plug and Play (UPnP) is a set of networking protocols promulgated by the UPnP Forum. The goals of UPnP are to allow devices to connect seamlessly and to simplify the implementation of networks in the home (data sharing, communications, and entertainment) and in corporate environments for simplified installation of computer components. UPnP achieves this by defining and publishing UPnP device control protocols (DCP) built upon open, internet-based communication standards.

We will be using Jinzora, the PHP-based media server for this specific guide.

11.1 Installing Jinzora Media Server

Jinzora is a web-based media streaming and management system. It is supported in various platforms and it allows you to access your music collection from any devices over the internet. It supports a variety of audio and video formats and even has support for on-the-fly transcoding. Jinzora needs LAMP to be installed on your system, this is dead simple in distributions like Ubuntu, where you can simply go to the synaptics package manager, select packages by task and choose LAMP from there.

One small plugin would be needed in the P or the PHP part of the above, for that simply do the following in your terminal:

```
$ sudo apt-get install php5-gd
$ sudo /etc/init.d/apache2 restart
```

The first command installs the plugin, while the second simply restarts the service.

Next open your `php.ini` or the configuration file to optimise your PHP service and get ready for jinzora, it would be located at `/etc/php5/apache2/php.ini` : You will need root permissions to open this.

Make the following changes in it:

Change php.ini setting

Some of the default setting in the php.ini did not meet the requirement of Jinzora2. We have to change it in order for Jinzora to work.

```
gksu gedit /etc/php5/apache2/php.ini
```

Search for

```
memory_limit = 16M ; Maximum amount of memory a script may consume (16MB)
```

and change the value to `64`. If it's already at 128, don't bother.

Search for

```
max_execution_time = 30 ; Maximum execution time of each script, in seconds
```

and change the value to `300`

Search for

```
post_max_size = 8M
```

and change the value to `32`

Search for

```
upload_max_filesize = 2M
```

change the value to `32`

Issue the following command again to account for the changes above

```
$ sudo /etc/init.d/apache2 restart
```

Now we will look at installing Jinzora in Ubuntu, in case you want to try it once before installing you can see a demo at **http://live.jinzora.org/**

First things first, you will need to download the the lastest and greatest release for your platfrom, in our case a Debian-based Linux environment, so point your browsers to

http://github.com/jinzora/jinzora3/downloads

Download the required file in either zip or tar.gz, we would recommend the latter due to the smaller size.

Next step would be to extract the contents of Jinzora3 folder to your home.

Fire up your terminal as always and copy the Jinzora3 folder to the root of the web server. You may need to adjust the name of the folder if it changes by the time this guide reaches you.

```
sudo cp jinzora-3.0 /var/www
sudo cp -r jinzora-3.0/var/www
```

The second command is needed to copy the directory itself, calling the recursive method, as the first one would have it ommitted it by default.

Run the configure script to change the file permission, refer to the second appendix of this guide to get a better understanding of the configure script.

```
cd /var/www/jinzora-3.0
```

```
sudo sh configure.sh
```

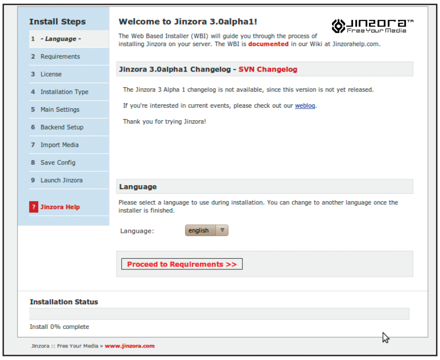
At this point you will be met with the following message

`You are now in setup mode.`

`Please direct your web browser to the directory where you installed Jinzora and load index.php - you will then be taken through the complete setup.`

To proceed from here, Open your browser, point the URL to **http://localhost/jinzora3.0**

You will be greeted with the Jinzora Welcome screen. If you get a 403 error message, just use `chmod` to adjust the permissions.



Welcome to Jinzora 3.0alpha1!

The Web Based Installer (WBI) will guide you through the process of installing Jinzora on your server. The WBI is documented in our Wiki at Jinzorahelp.com.

Jinzora 3.0alpha1 Changelog - SVN Changelog

The Jinzora 3 Alpha 1 changelog is not available, since this version is not yet released.

If you're interested in current events, please check out our weblog.

Thank you for trying Jinzora!

Language

Please select a language to use during installation. You can change to another language once the installer is finished.

Language: english

Installation Status

Install 0% complete

Jinzora :: Free Your Media » www.jinzora.com

The Jinzora welcome screen

Below the page, click the `Proceed to Requirements`‘ to continue the installation.

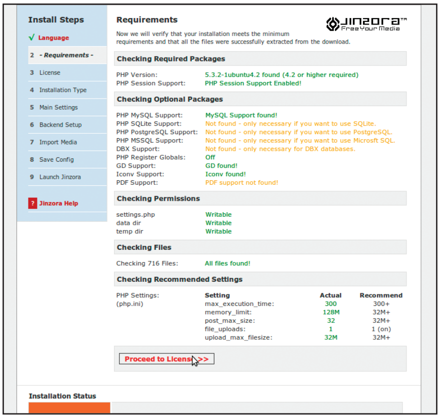
Jinzora will now check your system to see if it meets the minimum requirement. We have already taken care of this part, and you shouldn't ideally be facing any troubles over here.

Click the `Proceed to License` to continue.

This will bring you to the license agreement page. Check the box and click the `Proceed to Install Type`.

In the Installation Type dropdown box, select Standalone. Under the Jukebox Mode, choose `Streaming Only`. If you plan on running a server side Jukebox utilising MPD, select Jukebox and Streaming.

Click `Proceed to Main Settings` to continue.



Requirements

Now we will verify that your installation meets the minimum requirements and that all the files were successfully extracted from the download.

Checking Required Packages

```
PHP Version:              5.3.2-1ubuntu4.2 found (4.2 or higher required)
PHP Session Support:      PHP Session Support Enabled!
```

Checking Optional Packages

```
PHP MySQL Support:        MySQL Support found!
PHP SQLite Support:       Not found - only necessary if you want to use SQLite.
PHP PostgreSQL Support:   Not found - only necessary if you want to use PostgreSQL.
PHP MSSQL Support:        Not found - only necessary if you want to use Microsft SQL.
DBX Support:              Not found - only necessary for DBX databases.
PHP Register Globals:     Off
GD Support:               GD found!
Iconv Support:            Iconv found!
PDF Support:              PDF support not found!
```

Checking Permissions

```
settings.php              Writable
data dir                  Writable
temp dir                  Writable
```

Checking Files

```
Checking 716 Files:       All files found!
```

Checking Recommended Settings

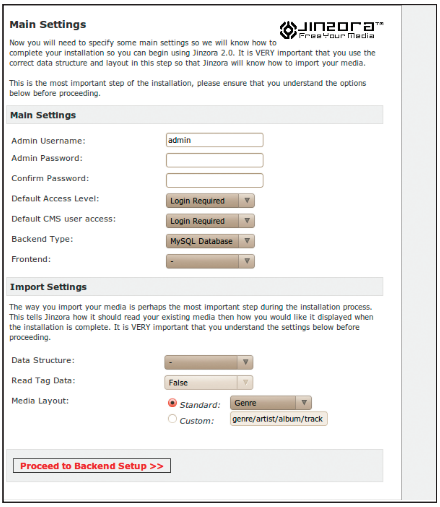
```
PHP Settings:             Setting                   Actual    Recommend
(php.ini)                 max_execution_time:        300        300+
                          memory_limit:              128M       32M+
                          post_max_size:             32         32M+
                          file_uploads:              1          1 (on)
                          upload_max_filesize:       32M        32M+
```

Installation Status

Setting the requirements of Jinzora

Enter the required details as asked and demonstrated above in the screenshot. You don't need to make any changes over here, for your Backend ensure it is MySQL, for your frontend you can choose what you wish.

Import Settings define how Jinzora handles your media tracks. If you have been diligently organising your audio files in proper folder order, select `Filesystem` under Data Structure. On the other hand, if you have carefully entered the meta data for each track, but all the tracks are jumbled up in one folder, select `Tag data` under Data Structure. Under the Media layout, select how you want Jinzora to display your media files – by Genres, Albums or Artists.



Main Settings

Now you will need to specify some main settings so we will know how to complete your installation so you can begin using Jinzora 2.0. It is VERY important that you use the correct data structure and layout in this step so that Jinzora will know how to import your media.

This is the most important step of the installation, please ensure that you understand the options below before proceeding.

Main Settings

Admin Username:	admin
Admin Password:	
Confirm Password:	
Default Access Level:	Login Required
Default CMS user access:	Login Required
Backend Type:	MySQL Database
Frontend:	-

Import Settings

The way you import your media is perhaps the most important step during the installation process. This tells Jinzora how it should read your existing media then how you would like it displayed when the installation is complete. It is VERY important that you understand the settings below before proceeding.

Data Structure:	-
Read Tag Data:	False
Media Layout:	[✓ Standard: Genre] Custom: genre/artist/album/track

After you have entered the requirements, proceed to licencing

Once done, click `Proceed to Backend Setup` to proceed.

In the backend setup, enter your MySQL username and password in the `Database User` and `Database password` fields (this is the same data as the one you entered when you installed LAMP on your system). Give your database a name (In this case, we labelled it `jinzora2)`. Make sure that the Database Server is set to `Localhost`, Database Type is `MySQL` and Create database is set to `True`. This is important! If you did not set the Create Database to True, Jinzora will not create a database and the whole installation will fail. Click `Proceed With Backend Install`.

At this point Jinzora will attempt to install the database, cross your fingers and hope for the best. If all works out then yours will be prompted with a Create Database successful screen. Now you just need to click on `Proceed to import Media`, to well import the media.

In the Media Directory field, enter the file path to your media files. This is almost like attaching a file. You can keep adjusting these settings later also, finally just click on `Import Media`.

Depending on the number of media files you have, this importing process might take a long time. If you still have media files scattered in other locations, enter the file path and click `Import Media`, else, click `Proceed to Save Config`. You have come to the end of Jinzora installation. Click `Proceed to launch Jinzora` to launch Jinzora.

Happy streaming.

Appendix A

A Linux system administrator who doesn't know shell scripting is like batman without BatMobile, with the Bat suite you will still like one, but that is about it. This is a quick and handy introduction to Bash scripting, it is written with the assumption that you have no previous knowledge of bash scripting.

Hello World and Variables

Open your favourite text editor (use vi, emacs, gedit or your own version) and save the follwing code in a file and name it as you fancy, but with a `*.sh` extension.

Code

```
  #!/bin/bash
  # declare a variable by the name of outP
  outP="Hello World"
  # displaying the variable on a screen
  e c h o                                          $ o u t P
#    dipslyaing    a    literal    string    on    screen
echo 'hello digit readers'
```

Explanation

`#!` stands for `shebang`. It informs the system about the interpreter for which this script is intended to run – in this case `Bash` located in the `/bin/bash` location. # as you must have guessed by now, indicates a comment, and only in the first line as a part of shebang symbol does it have special meaning.

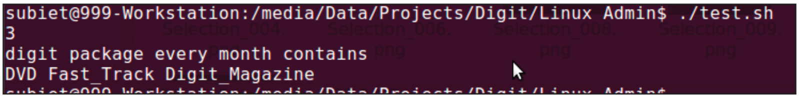
Next comes declaration of the variable, now it may seem natural to Python and like programmers, but people coming from C/C++ backgrounds should take note that neither do you need to declare a variable before using it nor do you need to specify a type. The type of variable is automatically determined by the content and how it is treated.

A variable is referred as `$Var_Name`, without the quotes, in the print syntax to distinguish it from a immutable string which will be represented by enclosing in quotes. The print snytax is the echo command. `Echo` is equivalent to `print` command in Python, `cout` in c++ and `printf` in C.

We will end this introduction after briefly mentioning about Array Variables. You can consider the following example which will illustrate declaring and accessing array varialbes and individual elements inside it.

```
  #!/bin/bash
  #Array declaration of elements
  Digit_Contents=( 'DVD' 'Fast_Track' 'Digit_Magazine' )
```

```
    # Displaying No. Elements in array
    echo ${#Digit_Contents[@]}
    # Display individual elements
    echo 'digit package every month contains'
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
3
digit package every month contains
DVD Fast_Track Digit_Magazine
```

Introduction to arrays

```
    echo ${Digit_Contents[0]} ${Digit_Contents[1]} ${Digit_
Contents[2]}
```

Just the variable name `$Digit_Contents` always refer to the first element is equivalent to writing `${Digit_Contents[0]}`.

Please note that the names are case sensitive, so Digit is not equal to digit.

Executing and quotes

As a first step towards executing your script you would need to explicitly grant it executalbe permissions. Some modern distributions after noticing a shebang symbol might automatically store it with executable permissions. Just to be sure, do the following in the directory where your script is stored.

```
$ chmod +x script_name.sh
```

To execute this script, issue the following at the prompt

```
$ ./script_name.sh
```

the / symbol is required to tell the shell that it needs to interpret/execute the script and not look for a program to open the file.

It is imperative to note that each command in the bash script can individually be executed sequentially on the shell also, because it is an interpred language. So the question is can bash commands be also executed from withing a script, and the answer is very loud yes because it is a very frequently used and useful feature. Executing shell commands with bash can be archived with backticks, to illustrate it, consider the following example

```
#!/bin/bash
# use backticks " ` ` " to execute shell command
echo `ls -l`
# executing bash command without backticks
echo
echo
echo ls -l
```

In the above example the first echo, would perform the same way as issuing the `ls -l` command on the prompt. The last echo would do nothing more than just print `ls -l` exactly on the string. So the question is why did we use two blank echo commands in the middle? That was a simple way of inserting new line characters.

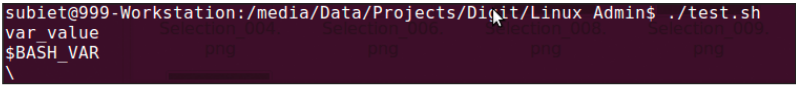
If you notice that `echo 'hello'"`, `"echo "hello""` and a `echo hello` seem to be producing the same results, so let us have a little detailed look at it since quotation mark are an extremely important and confusing feature of scripting. We will deal with the meta characters or special characters. Let us take the example where we want to use the `$` character as just the character and not with its special meaning.

This will be illustrated nicely by the following example

```
#!/bin/bash

#Declare bash string variable
var_name="var_value"
# echo variable var_name
echo $var_name
#when meta character such us "$" is escaped with "\" it
will be read literally
echo \$BASH_VAR
# Since \ is also a meta character its meaning can be
suppressed with another \
echo "\\"
```

Single quotes can also be used to suppress the special meaning of any meta characters, so the above example can also be re-written with single quotes. It is not possible to use another single quote within two single quotes not even if the single quote is escaped by backslash.

```
#!/bin/bash

#Declare bash string variable
var_name="var_value"
# echo variable var_name
echo $var_name
#when meta character such us "$" is escaped with "\" it
will be read literally
echo '$BASH_VAR'
# Since \ is also a meta character its meaning can be
suppressed with another \
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
var_value
$BASH_VAR
\
```

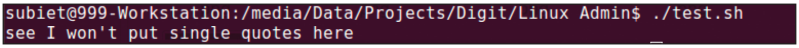
The output of the code above

```
echo '\'
```

both the above will produce the following output:

Moving on to Double quotes, they will suppress special meaning of almost every meta characters except $, \ and `. Rest of all the meta characters will be read literally. It is also possible to use single quote within double quotes.

```
#!/bin/bash
#use of single quotes within double quotes
echo "see I won't put single quotes here"
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
see I won't put single quotes here
```

Without single quotes

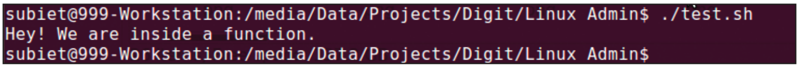
will produce the following:

Functions and More

A function is declared by using the `function` keyword in shell scripting followed by the function name.

```
#!/bin/bash
# Declaring a function
function digit_func
{
     echo 'Hey! We are inside a function.'
}
digit_func
```

Please note the brackets after the fuction name, which in most programming languages enclose the parameter list. This is because the



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
Hey! We are inside a function.
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$
```

Your first function

paramaters are by default assigned variable names as `$1`, `$2` and so on while `$0` will refer to the script name by default. This greatly increases the speed of programming at the cost of slight loss of flexibility.

Parameters are passed to a function by merely separating them with a space next to the function call. To illustrate what we are saying, consider this:

```
#!/bin/bash
# Declaring a function
function digit_func
{
     echo 'Hey! We are inside a function.'
     echo $1
```

```
        echo $2
    }
    digit_func 'First_Argument'
    digit_func 'First_Argument' 'Second Argument'
```

produces the output:

```
    Hey! We are inside a function.
    First_Argument

    Hey! We are inside a function.

First_Argument

Second Argument
```

since, there is no second argument in the first call, `$2` is blank, and then `echo $2` command merely works a newline.

Dealing with command line parameters

It is done in much the same way as arguments to a function. For example on issuing the following command

```
    $ ./script_name.sh param1 param2 param3
```

`param1` would be assigned variable name `$1`, `param2` as `$2` and `param3` as `$3` and here is how we can deal with them. We would make a simple script that would delete a single file, which can be passed on as a command line argument. `rm` command standing for remove is used in Linux for deleting the file, if you find it difficult to remember, create a script called `./delete.sh` and store it with the following contents:

```
    #!/bin/bash
    # Declaring a function
    echo 'deleting file $1'
    echo `rm $1`
    echo 'deleted the file $1'
```

and now if you execute the following in the shell:

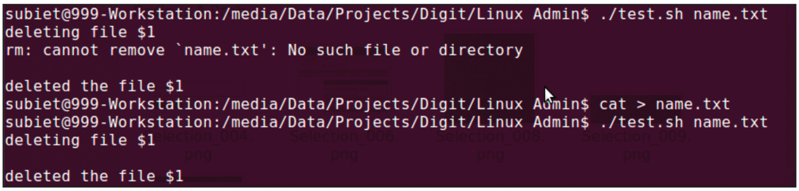
```
    $ ./delete.sh file_to_be_deleted.txt
```

it will just do the job.

User-Input, Bash Trap and If-Else

Till now we haven't considered any example where we had to take input from a user, except for the one passed as command line parameters. User input is primarily handled by the `Read` command. The following example should illustrate it well.

```
   $#!/bin/bash
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh name.txt
deleting file $1
rm: cannot remove `name.txt': No such file or directory

deleted the file $1
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ cat > name.txt
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh name.txt
deleting file $1

deleted the file $1
```

Command Over Loading :"

```
   echo "I am ready, give me some input"
   read  bored
   echo    "The    input    you    gave    was    $bored"
#Now  we  will  illustrate  usage  of  the  defaul  'REPLY'
variable
   echo    "Ok,    Now    give    me    something    more"
echo "My shell remembers what you entered, it was $REPLY"
```

There isn't really much more to input in bash, it is simple, elegant and fast, so let us move to something interesting, the `Bash Trap` command. This command will let you handle `[Ctrl] + [C]` soft user interrupt in a more polished manner. Take the example below:

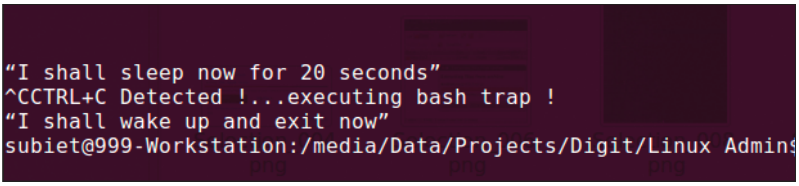
```
   #!/bin/bash
   # bash trap command
   trap bashtrap INT
   # bash clear screen command
   clear;
   #  Also  please  notice,  how  we  are  executing  system
commands without the backticks.

   #  bashtrap  is  default  declared  functon,  we  are  now
defining it here.
   bashtrap()
   {
        echo  "CTRL+C  Detected  !...executing  bash  trap  !"
     echo "I shall wake up and exit now"
       exit
   }
```

```
  # Delay command illustrated before, the number indicates
seconds
  echo “I shall sleep now for 20 seconds”
  sleep 20
  echo "Exit Bash Trap Example!!!"
```

In the above example, we have illustrated the usage of clear and sleep command also.

```
  If - else - fi
```

We will begin with an example, that too with a nested `if - else -fi`, it is extremely simple to understand, we have heavily commented the code for better understanding.



```
“I shall sleep now for 20 seconds”
^CCTRL+C Detected !...executing bash trap !
“I shall wake up and exit now”
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$
```

Bash trap is as important as unimportant you currently think it is.

Consider the following example where we input a user name and display the password of the user, if the file `username.txt` exists that contains the password of the user, else we give the option to the user to create that file and store it with password for future refernce.

```
  #!/bin/sh

  # Prompt for a user name...
  echo "Please enter your name:"
  read USERNAME

  # Check for the file. We are using the -s option here,
more will be illustrated # later

  if [ -s ${USERNAME}.txt ]; then
          # Read the password from the file.
          passwd=`cat ${USERNAME}.txt`
          echo "Your stored password is $passwd"
  else
          # Asking if he wants to create it
```

```
             echo "Do you want to store the password now
(Y/N)"
           read answer

           if [ "$answer" = 'n' ]; then
                echo "Ok, as you wish, I was just helping"
           else
                     if [ "$answer" = 'N' ]; then
                               echo "Fine by me, we won't
store it"
                     else
                           # Write the age to a new file.
                           echo $answer > ${USERNAME}.txt

                    #fi is used to indicate the closing of
if.
                     fi
           fi
  fi
```

A test run of the above produced the following output

```
  Please enter your name:
  Digit
  Do you want to store the password now (Y/N)
  n
  Ok, as you wish, I was just helping
```

The above could obviously have been done in a slightly more efficent and far less dramatic manner, but we just wanted to illustrate the use of nested `if-else-fi`. Note unlike Python, we're not using indentation for forming the code blocks, we are just using them as pretty type. Code blocks are explicitly defined as the space between `if and fi` or `else and fi`.

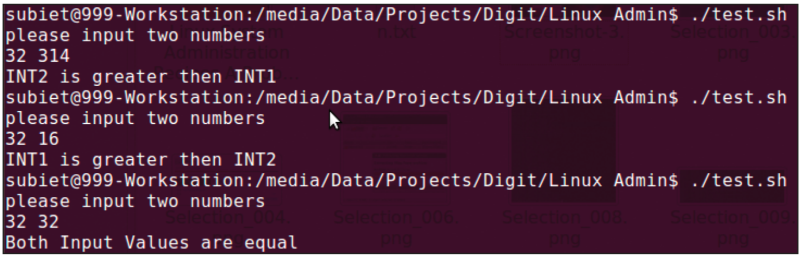
Comparison and File Handling

Arithmetic

The `if` command described above is hardly useful without some comparison going on, in the example above we have used string comparison which we will elaborate on further, currently let us have a look on some integer comparison

```
  -lt
  <
  -gt
```

```
>
-le
<=
-ge
>=
-eq
==
-ne
!=

#!/bin/bash
# Variable Declaration
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
please input two numbers
32 314
INT2 is greater then INT1
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
please input two numbers
32 16
INT1 is greater then INT2
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
please input two numbers
32 32
Both Input Values are equal
```

Output for the second program

```
INT1=2
INT2=2
if [ $INT1 -eq $INT2 ]; then
        echo "Both Values supplied are equal"
else
        echo "Values supplied are NOT equal"
fi
```

The above will always result in the following output below, courtesy the fixed values, in the next example we will take user input values

```
$ Both Values supplied are equal

#!/bin/bash
# Requesting use for input variables
echo please input two numbers
read INT1 INT2

if  [ $INT1 -eq $INT2 ]; then
        echo "Both Input Values are equal"
```

```
  #elif is used to continue the if-else-fi ladder for more
comparisons
  #this is NOT nested if.
  elif [ $INT1 -gt $INT2 ]; then
          echo "INT1 is greater then INT2"
  else
          echo "INT2 is greater then INT1"
  fi
```

Having taken a look at how to deal with numbers, let us move to the second major class that is, string comparisons, following the symbol look up table, just as is the case with arithmetic operators

```
  =
  equal
  !=
  not equal
  <
  less then
  >
  greater then
  -n s1
  string s1 is not empty
  -z s1
  string s1 is empty

  #!/bin/bash
  #Read string Str1 from the user
  read Str1
  #Read string Str2 from the user
  read Str2
  if [ $Str1 = $Str2 ]; then
          echo "Both Strings are equal"
  else
          echo "Strings are NOT equal"
  fi
```

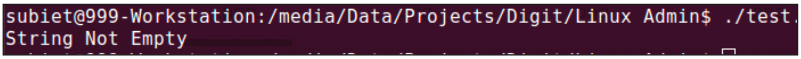
Gives the following output:

```
  subiet@999-Workstation:/media/Data/Projects/Digit/Linux
Admin$ ./test.sh
  case
  case
  Both Strings are equal
  subiet@999-Workstation:/media/Data/Projects/Digit/Linux
Admin$ ./test.sh
  case1
  case2
  Strings are NOT equal
```

Let us take a look at another example of string comparison, this time checking for empty strings

```
  #!/bin/bash
  #Declare string Str1
  Str1=""
  if [ -n $Str1 ]; then
          echo "String is as Empty as it could be"
  fi
```

We will now just provide the lookup table for files, out of which '-s'



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.
String Not Empty
```

String Comparison Examples

example we already saw in the nested if-else-fi example, where we used it to check for the existence of file. Using the rest is analogous to that.

```
  -b filename
  Block special file
  -c filename
  Special character file
  -d directoryname
  Check for directory existence
  -e filename
  Check for file existence
  -f filename
  Check for regular file existence not a directory
  -G filename
  Check if file exists and is owned by effective group ID.
  -g filename
  true if file exists and is set-group-id.
  -k filename
  Sticky bit
  -L filename
  Symbolic link
  -O filename
  True if file exists and is owned by the effective user
id.
  -r filename
  Check if file is a readable
  -S filename
  Check if file is socket
  -s filename
  Check if file is nonzero size
  -u filename
  Check if file set-ser-id bit is set
  -w filename
  Check if file is writable
  -x filename
  Check if file is executable
```

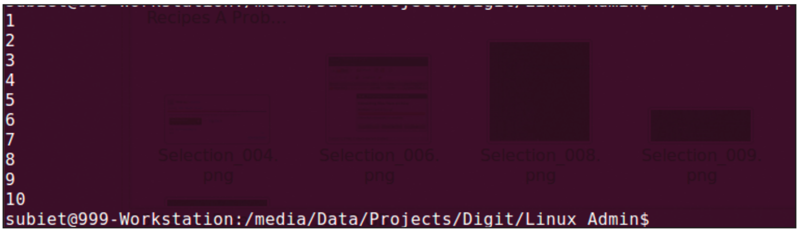
Loops, select and case

We primarily looked at conditional flow of our script, we have discussed input and output before that in sufficient detail, now the last important part that remains is iterative control of your script, for that we present 4 methods namely:

```
  1.for
  2.while
  3.select
  4.case
```

We will start with a personal favourite, `for` and then move on to others:

```
  FOR:
  Let us begin with a simple example and then we
  #!/bin/bash
  # bash for loop
  # notice it is backticks and not single quotes
```



```
1
2
3
4
5
6
7
8
9
10
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$
```

Loopy Loops

```
for i  in `seq 1 10`; do
     echo $i
done
```

The above self explanatory code example produces the output as follows:
Let us overload the `ls` command to display the information in a little more verbose mode, we will again use the command line parameters to get the feel of the command.

```
#!/bin/bash
      for i in $( ls $1 ); do
          echo "List of files in proc are  $i"
      done
```

so now on issuing the following command, assuming the script name is `list_files`.

```
$ ./list_files.sh /proc/
```

the output will be as follows:

```
...
List of files in proc are  swaps
List of files in proc are  sys
List of files in proc are  sysrq-trigger
List of files in proc are  sysvipc
List of files in proc are  timer_list
List of files in proc are  timer_stats
List of files in proc are  tty
List of files in proc are  uptime
List of files in proc are  version
List of files in proc are  version_signature
List of files in proc are  vmallocinfo
List of files in proc are  vmstat
List of files in proc are  zoneinfo
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$
```

The `for` in the shell operates like Python. Let us take the example of directory listing.

WHILE

Moving on the `while` loop we will again try and demonstrate similar functionality to bring out a comparison.

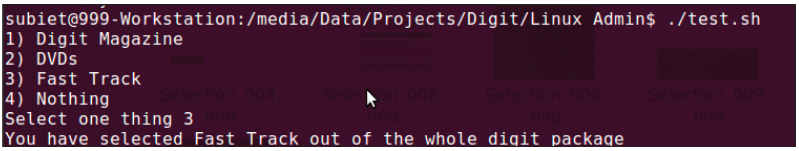
```
#!/bin/bash
loop_cnt=10
# bash while loop
while [ $loop_cnt -gt 0 ]; do
        echo Value of loop_cnt is: $loop_cnt
        let loop_cnt=loop_cnt-1
done
```

gives the output as:

```
Value of loop_cnt is: 10
Value of loop_cnt is: 9
Value of loop_cnt is: 8
Value of loop_cnt is: 7
Value of loop_cnt is: 6
Value of loop_cnt is: 5
Value of loop_cnt is: 4
Value of loop_cnt is: 3
Value of loop_cnt is: 2
Value of loop_cnt is: 1
```

We have skipped the `Until` loop which is very similar to the while loop, except for checking of initial condition. Now we have covered two loops, let us take a look at select and case, again illustrating via an example should familiarise you quickly with these useful tools.

```
#!/bin/bash
# PS3 is a voodoo word, don't disturb. I am serious.
PS3='Select one thing '
# bash select example
select content in "Digit Magazine" "DVDs" "Fast Track"
"Nothing"
do
   echo "You have selected $content out of the whole
digit package"
  # Break the infinite loop.
```



```
subiet@999-Workstation:/media/Data/Projects/Digit/Linux Admin$ ./test.sh
1) Digit Magazine
2) DVDs
3) Fast Track
4) Nothing
Select one thing 3
You have selected Fast Track out of the whole digit package
```

Illustration of select

```
    break
  done
  # exit with zero return value to OS, indicating
successful termination.
  exit 0
```

Notice how the numbering is added automatically for you.

Next we provide a case example, look at how it ends with the word esac, which is the inverse of case, just like `fi` for `if`.

```
  #!/bin/bash
  echo "What is your preferred thing in the Digit Package"
  echo "1) Fast Track"
  echo "2) Obviously Fast Track"
  echo "3) No Questions, Fast Track"
  echo "4) Read option 1"
  echo "5) Digit, What?"
  read case;
  #simple case bash structure
  # note in this case $case is variable and does not have
to
  # be named case this is just an example
  case $case in
      1) echo "You selected the correct option";;
      2) echo "Right Choice";;
      3) echo "Good Job";;
      4) echo "You are correct, as always!";;
            5)   echo  "Are   you   on   Windows   3.1";;

  esac
```

produces the following as two test runs:

```
  subiet@999-Workstation:/media/Data/Projects/Digit/Linux
Admin$ ./test.sh
  What is your preferred thing in the Digit Package
  1) Fast Track
  2) Obviously Fast Track
  3) No Questions, Fast Track
  4) Read option 1
  5) Digit, What?
  2
  Right Choice
  subiet@999-Workstation:/media/Data/Projects/Digit/Linux
```

```
Admin$ ./test.sh
  What is your preferred thing in the Digit Package
  1) Fast Track
  2) Obviously Fast Track
  3) No Questions, Fast Track
  4) Read option 1
  5) Digit, What?
  5
  "Are you on Windows 3.1"
```

Appendix B

Commonly you would find that the application/software packacge that you wish to install is not available as a deb/rpm or as binary package of your choiced distribution or your platform, in that event your only option is to install from source. Installing from source all has certain advantages, which shall be especially crucial for an advanced Linux user, the primary being the fact that application is optimised for your machine and it performs in more stable and fast manner, since while compiling and linking it takes in the parameter for your specific machine into account to generate the final bytecode. On the downside, installing from source requires you to satisfy a lot more dependencies, most of which you will have to manually install. For example, if you require libXYZ for an application to work, you would also require libXYZ-dev for it compile on your machine.

Before we proceed any further, let us prepare our machine for the task by installing a few common necessities.

```
$ sudo apt-get install build-essential checkinstall
```

followed by

```
$ sudo apt-get install cvs subversion git-core mercurial
```

The first command is for the necessary tools for compiling and linking from source, the second one is for installing the tools that are required for obtaining the source code.

Now the first step towards installing applications from source is to obtain the source code. There are two popular ways to get it, one as a simple compressed archive download from the project web page on the internet and the second is to download it from the version control system. The second method delivers more up-to-date packages and once you get a hang of it, it is also the easier way.

If you have downloaded the package from the internet as a compressed archive, you would need to extract the contents. Typically the packages will be a tar.gz file. We are listing the instructions for tar.gz, tar.bz2 and zip, which should cover 99.5 per cent of all packages found.

For tar.gz

```
$> tar -zxvf <filename>
```

For tar.bz2

```
$>  tar -jxvf <filename>
```

For .zip

```
$>  unzip <filename>
```

All the three above methods will extract the content of the package into the current directory. So ensure you are in the directory where you want to dump the source code. Kindly note, this has nothing to do with the installation directory that will be decided automatically by the installer.

A good practice would be to create a directory to put all your source code in, the recommended place is to use `/usr/local/src`, you should ensure that you have write privelleges for that directory, this can also be done by the following

```
$ sudo chown your_user_name /usr/local/src
$ sudo chmod u+rwx /usr/local/src
```

As you might have guessed, you can replace `/usr/local/src` with whatever directory you wish, but the default is the recommended way.

You can also use the graphical way, if you have the option of, just using an archive utility and extracting them into a location of your choice

Moving to the second method of obtaining the source code, or the version control way. There are four popular revision controls used for open-source software

- SVN
- GIT
- CVS
- Mercurial

The commands above have already installed the required tools for using them, on the project page there would be an explicit command for their respective version control, in our htop example it is:

```
$ svn co https://htop.svn.sourceforge.net/svnroot/htop/
trunk htop
```

Once you have obtained the package it is always a good idea to look for a ReadMe.txt or a install.txt, which will typically contain helpful instructions towards compiling and installing the software on your machine. They may also inform you about any special requirements that are there, if any.

Now depending on how the file is packaged you may have multiple options, the most common option is to look for a `configure` file in the contents of the source code you have just downloaded. If present, then open a terminal, switch to the directory where you have extracted the source code, and type in

```
$ ./configure.
```

You can also run `./configure` with certain parameters to further customise your install, the details of which will be available only in the `ReadMe.txt` and `install` file, and will typically differ from each application to another. A good shot in the dark though may be to run the following to get a quick help

```
$ ./configure --help
```

Now if all goes well, the result of the `./configure` command will make your system ready for application install (it will display a success message), but that is often not the case, which leads us to our next step of installing dependencies

We will now install all the missing dependencies manually.

To prepare, install the package apt-file, if not already installed, this can be done via

```
$ sudo apt-get install apt-file
```

and then

```
$ sudo apt-file update.
```

At this point due to the low verbosity of the above it may appear that nothing is actually happening, but please wait while the above finishes.

The apt-file program has some interesting functions, the two most useful are apt-file search which searches for a particular file name, and apt-file list which lists all the files in a given package.

Now refer to the output of the `./configure` command from above, if the output resembles something like `configure: error: Library requirements some_thing not met and_more_something_something`

But right above that it will list a filename that it cannot find (often a filename ending in `.pc`, for instance). So now you need to run the following command

`$ apt-file search missingfilename.pc` which informs you the Ubuntu package the missing file is in. Then to take care of that you can run the following

```
$ sudo apt-get install requiredpackage
```

Then try running `./configure` again, and see if it works. If you get to a bunch of text that finishes with `config.status: creating Makefile` followed by no obvious error messages, you're ready for the next steps, if not reiterate the process.

The final frontier

Congratulations, though you haven't finished yet, but you have covered the part mentioned above this, what lies next should be cake walk and rewarding too. In the same terminal session, or if you have closed your terminal than open it and go the same directory where you had extracted your source and issue the following

```
$ make
```

Mind you that for a large program this can be quite some time, for example during installation of Firefox, you can possibiliy finish your lunch with ease, of course setting the `CONCURRENCY_LEVEL` to the number of processors/ cores you have to speed things up a little.

This has compiled your program, with the assumption you actually want to install it since you have done so much work, issue the following:

```
$ sudo checkinstall
```

The above is a replacement for sudo make install that might be there in some tutorial or in the readme. It is a better way, and it also puts it in your package manager so you can use synaptic to remove it. Also this creates a deb file so you can distribute it for your friends and earn brownie points.

Yes, we agree, this ain't for the faint hearted but trying out a few times will make the process far smoothe than it looks currently.

Appendix C

System information commands

`df`

The df command displays filesystem and disk space usage for all partitions. The command `df-h` is probably the most useful. It uses megabytes (M) and gigabytes (G) instead of blocks to report. (`-h` means "human-readable.")

`free`

The free command displays the amount of free and used memory in the system. `free -m` will give the information using megabytes.

`top`

The top command displays information on your Linux system, running processes, and system resources, including the CPU, RAM, swap usage, and total number of tasks being run.

`uname -a`

The `uname` command with the `-a` option prints all system information, including machine name, kernel name, version, and a few other details.

`lsb_release -a`

The `lsb_release` command with the `-a` option prints version information for the Linux release you're running. For example:

```
  user@computer:~$ lsb_release -a
  LSB Version:     n/a
  Distributor ID: Ubuntu
  Description:     Ubuntu (The Hardy Heron Release)
  Release:
  Codename:        hardy
```

`ifconfig`

This reports on your system's network interfaces.

`iwconfig`

The command iwconfig will show you any wireless network adapters and the wireless specific information from them, such as speed and network connected.

`ps`

The command ps allows you to view all the processes running on the machine.

The following commands list the hardware on your computer, either a specific type or with a specific method. They are most useful for debugging when a piece of hardware does not function correctly.

```
lspci
```
The command lspci lists all PCI buses and devices connected to them. This commonly includes network cards and sound cards.

```
lsusb
```
The command lsusb lists all USB buses and any connected USB devices, such as printers and thumb drives.

```
lshal
```
The command lshal lists all devices the hardware abstraction layer (HAL) knows about, which should be most hardware on your system.

```
lshw
```
The command lshw lists hardware on your system, including maker, type, and where it is connected.

Working with the file system

The commands which you use for moving around the file system include the following:

```
pwd
```
pwd stands for "print working directory. The pwd command will tell you the directory in which you're located.

```
cd
```
The cd command lets you change directories. When you open a terminal for the first time, you are in your home directory. So, to move around the file system you have to use cd. For example, cd ~/Desktop will move you to your desktop directory.

To navigate to the root directory, use `cd /`
To navigate to your home directory, use `cd~`
To navigate up one directory level, use `cd ..`
To navigate to the previous directory (or back), use `cd -`
To navigate through multiple levels of directory at once, use `cd / var/ www.` For example, which will take you directly to the `/www subdirectory of /var/`.

Manipulating Files and Folders

You can manipulate files and folders using the following commands:

`cp`

The cp command will make a copy of a file for you. For example, cp file foo will make an exact copy of file and name it "foo," but the file "file" will still be there. When you use mv, that file would no longer exist, but when you use cp the file stays and a new copy is made.

`mv`

The mv command will move a file to a different location or will rename a file. Examples are as follows: mv file foo will rename the file "file" to "foo." mv foo ~/Desktop will move the file "foo" to your desktop directory but will not rename it. You must specify a new file name to rename a file.

To save on typing, you can substitute ~ in place of the home directory.

Note

If you are using mv with sudo you will not be able to use the ~ shortcut. Instead, you will have to use the full pathnames to your files.

`rm`

Use this command to remove or delete a file in your directory. It will not work on directories in which there are files.

`ls`

The ls command will show you the files in your current directory. Used with certain options, you can see file sizes, when files were created, and file permissions. For example, ls ~ will show you the files that are in your home directory.

`mkdir`

The mkdir command will allow you to create directories. For example, mkdir music will create a music directory.

`chmod`

The chmod command will change the permissions on the files listed.

Permissions are based on a fairly simple model. You can set permissions for user, group, and world, and you can set whether each can read, write, and execute the file. For an example, if a file had permission to allow everybody to read but only the user could write, the permissions would read rwxr-r-. To add or remove a permission, you append a + or a - in front of the specific permission. For example, to add the capability for the group to edit in the previous example, you could type chmod g+x file.

`chown`
The chown command allows the user to change the user and group ownerships of a file. For example, chown jim file would change the ownership of the file to"Digit"

Searching and Editing Text Files
Search and edit text files using the following commands:

`grep`
The command grep allows you to search inside a number of files for a particular search pattern and then print matching lines. For example, grep blah file will search for the text "blah" in the file and then print any matching lines.

`sed`
The sed (or Stream EDitor) command allows search and replace of a particular string in a file. For example, if you wanted to find the string "cat" and replace it with "dog" in a file named pets, you would type sed s/cat/dog/g.

Three other commands that are useful for dealing with text include:

`cat`
The cat command, short for concatenate, is useful for viewing and adding to text files. The simple cat FILENAME will display the contents of the file. Using cat filename file adds the contents of the first file to the second.

`nano`
Nano is a simple text editor for the command line. To open a file, use nano filename. Commands listed at the bottom of the screen are accessed via ctrl+letter name.

`less`
The less command is used for viewing text files as well as standard output. A common usage is to pipe another command through less to be able to see all the output, such as `ls | less`.

Dealing with Users and Groups
You can use the following commands to administer users and groups:

`adduser`

The adduser command will create a new user. To simply create a new user, type sudo adduser $loginname. This will create the user's home directory and default group. It will prompt for a user password and then further details about the user.

`passwd`

The passwd command will change the user's password. If simply run by a regular user, it will change his password. If run using sudo, it can change any user's password. For example, sudo passwd digit will change Digit's password.

`who`

The who command will tell you who is currently logged into the machine.
addgroup: The addgroup command will add a new group. To create a new group, type sudo addgroup $groupname.

`deluser`

The deluser command will remove a user from the system. To remove their files and home directory, you need to add the -remove-home option

`delgroup`

The delgroup command will remove a group from the system. You cannot remove a group that is the primary group of any users.

Getting Help on the Command Line

This section will provide you with some tips on getting help on the command line. The commands -help and man are the two most important tools at the command line.

Virtually all commands understand the -h (or -help) option which will produce a short usage description of the command and its options, then exit back to the command prompt. Try man -h or man -help to see this in action.

Every command and nearly every application in Linux will have a man (manual) file, so finding them is as simple as typing man command to bring up a longer manual entry for the specified command. For example, man mv will bring up the mv (Move) manual.

Some helpful tips for using the man command include:

arrow keys

Move up and down the man file using the arrow keys.

`q`

Quit back to the command prompt by typing [Q].

```
man man
```

`man man` will bring up the manual entry for the man command, which is a good place to start!

```
man intro
```

`man intro` is especially useful. It displays the Introduction to User Commands, which is a well-written, fairly brief introduction to the Linux command line.

There are also info pages, which are generally more in-depth than man pages. Try info info for the introduction to info pages.

Searching for Man Files

If you aren't sure which command or application you need to use, you can try searching the man files.

```
man -k foo
```

This will search the man files for "foo." Try `man -k` nautilus to see how this works.

Note

`man -k foo` is the same as the apropos command.

```
man -f foo
```

This searches only the titles of your system's man files. Try man -f gnome, for example.

Note

`man -f foo` is the same as the whatis command.

Using Wildcards

Sometimes you need to look at or use multiple files at the same time. For instance, you might want to delete all .rar files or move all .odt files to another directory. Thankfully, there are series of wildcards you can use to acomplish this.

* will match any number of characters. For example, *.rar will match any file with the ending of .rar
* ? will match any single character. For example, ?.rar will match a.rar but not ab.rar
* `[characters]` will match any of the character within the brackets. For example, [ab].rar will match a.rar and b.rar but not c.rar

* `[!characters]` will match any characters that are not listed. For example, [!ab].rar will match c.rar but not a.rar or b.rar.

Hacking with Sudo

Sudo allows you to run a command with root privileges. The default installation of Ubuntu includes the Sudo command (sudo) and gives the default user account access to this command. If you need something to run as root, you can use sudo. A related command, sudoedit, allows you to edit a file as root. For example, to edit the root-only file `/var/spool/anacron/cron.daily`, you can use wither of the following commands:

```
sudo vi /var/spool/anacron/cron.daily
sudoedit /var/spool/anacron/cron.daily
```

Some history

When Unix was young (circa 1970), administrators needed a way to change between user access privileges. This was accomplished with the su command. Although originally designed to switch user access, it was mainly used to change privileges from a regular user to root. (The su command is sometimes called the super-user command for this reason.)

There are a couple of huge risks with using `su` to run commands as root. Most notably, all commands are executed with root privileges. Many times, only one command within a long list of commands really needs root access. Also, when a terminal is logged in as root, users may forget which user they are running as. Accidentally deleting files from a directory as root can be much more devastating than doing so as a regular user.

Sudo was created to limit the commands that can run as root. Rather than changing all privileges, only the privileges for a specific command are changed. Furthermore, Sudo can even limit which commands can be run as root. When the command completes, you are back to your regular user access. This way, only the required steps run as root, and you won't accidentally leave a terminal logged in a root.

Although very old Unix systems still rely on su to change access levels, it is being forcefully replaced by sudo. Furthermore, in most BSD and Linux distributions (including Ubuntu), Sudo has become standard for using root privileges. Although the su command is included with the Ubuntu installation, you cannot use it to run commands as root unless you first set a root password using sudo passwd root.

Adding Users to Sudo

The configuration file for sudo is `/etc/suders`. This lists who can run commands as root, which commands they can run, and any configuration preferences. Although this file can be edited using sudo `vi /etc/sudoers` or `sudoedit /etc/sudoers`, this is not recommended. Instead, the command `sudo visudo` (short for `vi sudo`) is the preferred approach. The visudo command ensures that only one person edits the file at a time. It also checks for syntax errors before installing the new file. (The last thing you need is to corrupt /etc/sudoers and lose the ability to run sudo for fixing the problem.)

The default `/etc/sudoers` file gives access to everyone who is part of the admin group. Non-admin users are blocked from running commands as root. If you want to add someone to the admin group, use:
1. Use the vigr command to edit the /etc/groups file.
2. sudo vigr
3. The /etc/groups file lists each group name, group ID number, and a comma-separated list of users. Search for the line that begins with admin. It should list one account name (your default Ubuntu account). For example, my default account name is neal and the line says:
4. admin:x:112:anot
5. Add the new user name to this line. Use a comma to separate usernames. For example, if I want to add the user marc to this line, I would have:
6. admin:x:112:anot,subeit
7. Save your changes and quit the editor.

If you want to give a user Sudo access without adding them to a group, then you can add them as a privileged user.

1. Edit the /etc/suderers file using sudo visudo.
2. Look for the line that says root ALL=(ALL) ALL.
3. Create a similar line with the user's account name. For example, if I want to give the account "jck" access, I will use:
4. root ALL=(ALL) ALL
5. jck ALL=(ALL) ALL

This addition says that the users "root" and "jck" can use sudo and can run any command with root privileges.
6. Save your changes and exit the visudo command.

Tweaking other Sudo Options

The default sudo command has three options enabled: !lecture, tty_tickets, and !fqdn. These options are flags-the ! in front of any flag disables the option. Other options can be added that include values. All the options belong on the Defaults line in the /etc/suders file (use sudo visudo to edit this file). Some command options that you might want to tweak include:

`lecture`

When this flag is enabled, the sudo command will display a warning about running commands as root. This can be useful if there are lots of administrators on a system. However, people usually ignore the warning. The default installation disables the lecture message: !lecture.

`timestamp_timeout`
This option lists the number of minutes before sudo requires you to re-enter your password. If you are running many sudo commands in a row, then it can become very inconvenient to re-enter your password after every command. The default configuration is 15 minutes (timestamp_timeout=15)- if your last sudo command was executed more than 15 minutes ago, then you will need to re-enter your password to run sudo. Setting the value to zero will always prompt for a password; setting the value to a negative number disables the timeout. If you are in a very sensitive environment, then you may want to lower this value (or set it to zero).

`tty_tickets`
If you have lots of windows open, then you will notice that you need to enter your password the first time you run sudo in any window. This is due to the tty_tickets flag-every terminal (tty) has its own sudo environment. If you disable this flag (!tty_tickets), then entering your password for sudo in one window will stop the sudo password prompts in all other windows.

`fqdn`
When host names are used in the sudoers file, this flag specifies the use of fully qualified domain names (fqdns).

`passwd_tries`
This sets how many password attempts the user gets before sudo fails. The default is passwd_tries=3.

`insults`
This is a fun flag. If the user enters in a bad password, then sudo will generate a random message. (The default is !insults.) For example:

```
$ sudo id
Password: [wrong]
Just what do you think you're doing Dave?
Password: [wrong]
It can only be attributed to human error.
Password: [wrong]
My pet ferret can type better than you!
sudo: 3 incorrect password attempts
```

Becoming Root

Although the Ubuntu security model has all administrative commands issued through sudo, there are some times when you really would be better off with a command prompt as root. Fortunately, there are many ways to accomplish this with sudo. A few examples:

```
   sudo -s             # run a shell as root
   sudo bash -o vi   # run a specific shell as root
   sudo -i            # set root's initial login environment
   sudo su - root     # become root and run root's login
environment
```

In the first two cases, the shell runs as root, but environment variables (for example, $HOME) are inherited. In the other two cases, the environment is replaced with root's real environment settings.

If you really need to be able to login as root, then you can use sudo passwd root to give the root account a password. As soon as you set the password, the root account can log in. You can always undo this change with sudo passwd -l root to lock the account.

Note

Enabling root logins is usually a bad idea. Logins give a trail of accountability. If someone logs in without a personalized account, then there is no way to identify the person who really logged in. sudo gives an audit trail in /var/log/auth.log (even though users with sudo access have the ability to blow away the logs).

Enabling Multiple CPUs (SMP)

Many of today's computers have multiple CPUs. Some are physically distinct, and others are virtual, such as hyper-threading and dual-core. In any case, these processors support symmetric multiprocessing (SMP) and can dramatically speed up Linux.

The kernel supports multiple CPUs and hyper-threading. If your computer has two CPUs that both support hyper-threading, then the system will appear to have a total of four CPUs.

There are a couple of ways to tell if your SMP processors are enabled in both the system hardware and kernel.

/proc/cpuinfo-This file contains a list of all CPUs on the system.

top-The top command shows what processes are running. If you run top and press 1, the header provides a list of all CPUs individually and their individual CPU loads. (This is really fun when running on a system with 32 CPUs. Make sure the terminal window is tall enough to prevent scrolling!)

System Monitor-The System Monitor applet can be added to the Gnome panels. When you click it, it shows the different CPU loads.

In each of these cases, if only one CPU is listed, then you are not running SMP. Multiple CPUs in the listings indicate SMP mode.

Disabling SMP

In some situations, such as application benchmarking or hardware debugging, you may want to disable SMP support. This can be done with the kernel parameters nosmp or maxcpus=1.If this is a temporary need, you can just boot the system, catch Grub at the menu by pressing ESC, and typing boot nosmp maxcpus=1 at the prompt. If you have multiple boot options, then you may need to edit the kernel line and add nosmp maxcpus=1 to the kernel boot line.

For a longer-term solution, consider adding these boot parameters to the Grub configuration.
1. As root, edit `/boot/grub/menu.lst`.
2. Scroll down to your kernel.
3. Add a kernel option for disabling SMP. For example:

```
kernel  /boot/vmlinuz-2.6.15-26-686  root=/dev/hda1  ro
splash maxcpus=1
```

5. Save your changes.

The next reboot will use your new changes.

Note: If you modify `/boot/grub/menu.lst`, be aware that the contents could be overwritten the next time you upgrade the kernel or run update-grub.

Missing SMP?

If you find that you only have one active CPU on a multiple CPU system, try installing the explicit SMP kernel: sudo apt-get install linux-686-smp. Beyond that, there are few generic debugging options and the problem is unlikely related to Ubuntu-it is probably a general Linux kernel problem.

Check with the motherboard manufacturer and see if Linux supports their chipset. For example, we have an old dual-CPU motherboard that is not supported by Linux.

Check the Linux Hardware FAQ for the motherboard or chipset. This will tell you if other people managed to get it to work. Web sites such as **https://wiki.ubuntu.com/HardwareSupport** and **http://www.faqs.org/docs/Linux-HOWTO/SMP-HOWTO.html** are good places to start.

If all else fails, post a query to any of the Linux or Ubuntu hardware forums. Maybe someone else knows a workaround. Some good forums include **http://www.ubuntuforums.org/**, **http://www.linuxhardware.org/**, and **http://www.linuxforums.org/forum/**. Be sure to include details such as the make and model of the motherboard, Ubuntu version, and other peripherals. It is generally better to provide too much information when asking for help, rather than providing too little.

Unfortunately, if SMP is not enabled after installing the linux-686-smp kernel, then it probably will not work. But you might get lucky-if someone has a patch then you will probably need to recompile the kernel.

Note

Compiling the kernel is not for the weak-of-heart. Many aspects of Linux have become automated or given easy-to-use graphical interfaces, but compiling the kernel is not one of them. There are plenty of help pages and FAQs for people daring enough to compile their own kernel.

Executing Multiple Commands

Often you may want to execute several commands together, either one after another or by passing output from one to another.

Run Sequentially

If you need to execute multiple commands in sequence, but don't need to pass output between them, you can run them using ; between each command. Each command will be executed, and the following command will be run. If you want to make the running of the second command conditional on the successful completion of the first command, separate the commands with `&&`.

If you need to execute multiple commands in sequence, but don't need to pass output between them, there are two options based on whether or not you want the subsequent commands to run only if the previous commands succeed or not. If you want the commands to run one after the other regardless of whether or not preceding commands succeed, place a ; between the commands. For example, if you want to get information about your hardware, you could run `lspci ; lsusb`, which would output information on your PCI buses and USB devices in sequence.

However, if you need to conditionally run the commands based on whether the previous command has succeeded, insert && between commands. An example of this is building a program from source, which is traditionally done with `./configure`, make, and make install. The commands make and make install require that the previous commands have completed successfully, so you would use `./configure && make && make install`.

Passing Output

If you need to pass the output of one command so that it goes to the input of the next, you need something called piping after the character used between the commands, `|` , which looks like a vertical bar or pipe.

To use the pipe, insert the `|` between each command. For example, using the `|` in the command `ls | less` allows you to view the contents of the ls more easily.